Network Working Group
Request for Comments: 4765
Category: Experimental

H. Debar
France Telecom
D. Curry
Guardian
B. Feinstein
SecureWorks, Inc.

March 2007

The Intrusion Detection Message Exchange Format (IDMEF)

Status of This Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Copyright Notice

IESG Note

The content of this RFC was at one time considered by the IETF, but the working group concluded before this work was approved as a standards-track protocol. This RFC is not a candidate for any level of Internet Standard. The IETF disclaims any knowledge of the fitness of this RFC for any purpose and in particular notes that the decision to publish is not based on complete IETF review for such things as security, congestion control, or inappropriate interaction with deployed protocols. The IESG has chosen to publish this document in order to document the work as it was when the working group concluded and to encourage experimentation and development of the technology. Readers of this RFC should exercise caution in evaluating its value for implementation and deployment.

Abstract

The purpose of the Intrusion Detection Message Exchange Format (IDMEF) is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems and to the management systems that may need to interact with them.

This document describes a data model to represent information exported by intrusion detection systems and explains the rationale for using this model. An implementation of the data model in the Extensible Markup Language (XML) is presented, an XML Document Type Definition is developed, and examples are provided.

Table of Contents

1. Introduction 4
1.1. About the IDMEF Data Model 4
1.1.1. Problems Addressed by the Data Model 5
1.1.2. Data Model Design Goals 6
1.2. About the IDMEF XML Implementation 7
1.2.1. The Extensible Markup Language 7
1.2.2. Rationale for Implementing IDMEF in XML 8
2. Notices and Conventions Used in This Document 10
3. Notational Conventions and Formatting Issues 10
3.1. IDMEF XML Documents 10
3.1.1. The Document Prolog 10
3.1.2. Character Data Processing in IDMEF 11
3.1.3. Languages in IDMEF 12
3.2. IDMEF Data Types 12
3.2.1. Integers 12
3.2.2. Real Numbers 12
3.2.3. Characters and Strings 13
3.2.4. Bytes 14
3.2.5. Enumerated Types 14
3.2.6. Date-Time Strings 14
3.2.7. NTP Timestamps 16
3.2.8. Port Lists 16
3.2.9. Unique Identifiers 17
4. The IDMEF Data Model and DTD 18
4.1. Data Model Overview 18
4.2. The Message Classes 20
4.2.1. The IDMEF-Message Class 20
4.2.2. The Alert Class 20
4.2.3. The Heartbeat Class 27
4.2.4. The Core Classes 29
4.2.5. The Time Classes 41
4.2.6. The Assessment Classes 42
4.2.7. The Support Classes 47
5. Extending the IDMEF 79
5.1. Extending the Data Model 79
5.2. Extending the IDMEF DTD 80
6. Special Considerations 81
6.1. XML Validity and Well-Formedness 81
6.2. Unrecognized XML Tags 82
6.3. Analyzer-Manager Time Synchronization 82
6.4. NTP Timestamp Wrap-Around 84
6.5. Digital Signatures 85
7. Examples 85
7.1. Denial-of-Service Attacks 86
7.1.1. The "teardrop" Attack 86
7.1.2. The "ping of death" Attack 87

7.2. Port Scanning Attacks 88
7.2.1. Connection to a Disallowed Service 88
7.2.2. Simple Port Scanning 89
7.3. Local Attacks 90
7.3.1. The "loadmodule" Attack 90
7.3.2. The "phf" Attack 93
7.3.3. File Modification 94
7.4. System Policy Violation 96
7.5. Correlated Alerts 98
7.6. Analyzer Assessments 99
7.7. Heartbeat 100
7.8. XML Extension 101
8. The IDMEF Document Type Definition (Normative) 104
9. Security Considerations 117
10. IANA Considerations 118
10.1. Adding Values to Existing Attributes 118
10.1.1. Attribute Registrations 119
10.1.2. Registration Template 130
10.2. Adding New Attributes and Classes 131
11. References 131
11.1. Normative References 131
11.2. Informative References 132
Appendix A. Acknowledgements 134
Appendix B. The IDMEF Schema Definition (Non-normative) 135

1. Introduction

The Intrusion Detection Message Exchange Format (IDMEF) [2] is intended to be a standard data format that automated intrusion detection systems can use to report alerts about events that they deem suspicious. The development of this standard format will enable interoperability among commercial, open source, and research systems, allowing users to mix-and-match the deployment of these systems according to their strong and weak points to obtain an optimal implementation.

The most obvious place to implement the IDMEF is in the data channel between an intrusion detection analyzer (or "sensor") and the manager (or "console") to which it sends alarms. But there are other places where the IDMEF can be useful:

- a single database system that could store the results from a variety of intrusion detection products would make it possible for data analysis and reporting activities to be performed on "the whole picture" instead of just a part of it;
- an event correlation system that could accept alerts from a variety of intrusion detection products would be capable of performing more sophisticated cross-correlation and cross-confirmation calculations than one that is limited to a single product;
- a graphical user interface that could display alerts from a variety of intrusion detection products would enable the user to monitor all of the products from a single screen, and require him or her to learn only one interface, instead of several; and
- a common data exchange format would make it easier for different organizations (users, vendors, response teams, law enforcement) to not only exchange data, but also communicate about it.

The diversity of uses for the IDMEF needs to be considered when selecting its method of implementation.

1.1. About the IDMEF Data Model

The IDMEF data model is an object-oriented representation of the alert data sent to intrusion detection managers by intrusion detection analyzers.

1.1.1. Problems Addressed by the Data Model

The data model addresses several problems associated with representing intrusion detection alert data:

- Alert information is inherently heterogeneous. Some alerts are defined with very little information, such as origin, destination, name, and time of the event. Other alerts provide much more information, such as ports or services, processes, user information, and so on. The data model that represents this information must be flexible to accommodate different needs.

  An object-oriented model is naturally extensible via aggregation and subclassing. If an implementation of the data model extends it with new classes, either by aggregation or subclassing, an implementation that does not understand these extensions will still be able to understand the subset of information that is defined by the data model. Subclassing and aggregation provide extensibility while preserving the consistency of the model.

- Intrusion detection environments are different. Some analyzers detect attacks by analyzing network traffic; others use operating system logs or application audit trail information. Alerts for the same attack, sent by analyzers with different information sources, will not contain the same information.

  The data model defines support classes that accommodate the differences in data sources among analyzers. In particular, the notions of source and target for the alert are represented by the combination of Node, Process, Service, and User classes.

- Analyzer capabilities are different. Depending on the environment, one may install a lightweight analyzer that provides little information in its alerts, or a more complex analyzer that will have a greater impact on the running system but provide more detailed alert information. The data model must allow for conversion to formats used by tools other than intrusion detection analyzers, for the purpose of further processing the alert information.

  The data model defines extensions to the basic Document Type Definition (DTD) that allow carrying both simple and complex alerts. Extensions are accomplished through subclassing or association of new classes.

- Operating environments are different. Depending on the kind of network or operating system used, attacks will be observed and reported with different characteristics. The data model should accommodate these differences.

  Significant flexibility in reporting is provided by the Node and Service support classes. If additional information must be reported, subclasses may be defined that extend the data model with additional attributes.

- Commercial vendor objectives are different. For various reasons, vendors may wish to deliver more or less information about certain types of attacks.

  The object-oriented approach allows this flexibility while the subclassing rules preserve the integrity of the model.

1.1.2. Data Model Design Goals

The data model was designed to provide a standard representation of alerts in an unambiguous fashion, and to permit the relationship between simple and complex alerts to be described.

1.1.2.1. Representing Events

The goal of the data model is to provide a standard representation of the information that an intrusion detection analyzer reports when it detects an occurrence of some unusual event(s). These alerts may be simple or complex, depending on the capabilities of the analyzer that creates them.

1.1.2.2. Content-Driven

The design of the data model is content-driven. This means that new objects are introduced to accommodate additional content, not semantic differences between alerts. This is an important goal, as the task of classifying and naming computer vulnerabilities is both extremely difficult and very subjective.

The data model must be unambiguous. This means that while we allow analyzers to be more or less precise than one another (i.e., one analyzer may report more information about an event than another), we do not allow them to produce contradictory information in two alerts describing the same event (i.e., the common subset of information reported by both analyzers must be identical and inserted in the same placeholders within the alert data structure). Of course, it is always possible to insert all "interesting" information about an

event in extension fields of the alert instead of in the fields where it belongs; however, such practice reduces interoperability and should be avoided whenever possible.

1.1.2.3. Relationship between Alerts

Intrusion detection alerts can be transmitted at several levels. This document applies to the entire range, from very simple alerts (e.g., those alerts that are the result of a single action or operation in the system, such as a failed login report) to very complex ones (e.g., the aggregation of several events causing an alert to be generated).

As such, the data model must provide a way for complex alerts that aggregate several simple alerts to identify those simple alerts in the complex alert's content.

1.2. About the IDMEF XML Implementation

Two implementations of the IDMEF were originally proposed to the Intrusion Detection Working Group (IDWG): one using the Structure of Management Information (SMI) to describe a Simple Network Management Protocol (SNMP) MIB, and the other using a DTD to describe XML documents.

These proposed implementations were reviewed by the IDWG at its September 1999 and February 2000 meetings; it was decided at the February meeting that the XML solution was best at fulfilling the IDWG requirements.

1.2.1. The Extensible Markup Language

The Extensible Markup Language (XML) [3] is a simplified version of the Standard Generalized Markup Language (SGML), a syntax for specifying text markup defined by the ISO 8879 standard. XML is gaining widespread attention as a language for representing and exchanging documents and data on the Internet, and as the solution to most of the problems inherent in HyperText Markup Language (HTML). XML was published as a recommendation by the World Wide Web Consortium (W3C) on February 10, 1998.

XML is a metalanguage -- a language for describing other languages -- that enables an application to define its own markup. XML allows the definition of customized markup languages for different types of documents and different applications. This differs from HTML, in which there is a fixed set of identifiers with preset meanings that must be "adapted" for specialized uses. Both XML and HTML use elements (tags) (identifiers delimited by '<' and '>') and attributes

(of the form "name=’value’"). But where "<p>" always means "paragraph" in HTML, it may mean "paragraph", "person", "price", or "platypus" in XML, or it might have no meaning at all, depending on the particular application.

NOTE: XML provides both a syntax for declaring document markup and structure (i.e., defining elements and attributes, specifying the order in which they appear, and so on) and a syntax for using that markup in documents. Because markup declarations look radically different from markup, many people are confused as to which syntax is called XML. The answer is that they both are, because they are actually both part of the same language.

For clarity in this document, we will use the terms "XML" and "XML documents" when speaking in the general case, and the term "IDMEF markup" when speaking specifically of the elements (tags) and attributes that describe IDMEF messages.

The publication of XML was followed by the publication of a second recommendation [4] by the World Wide Web Consortium, defining the use of namespaces in XML documents. An XML namespace is a collection of names, identified by a Uniform Resource Identifier (URI) [5]. When using namespaces, each tag is identified with the namespace it comes from, allowing tags from different namespaces with the same names to occur in the same document. For example, a single document could contain both "usa:football" and "europe:football" tags, each with different meanings.

In anticipation of the widespread use of XML namespaces, this memo includes the definition of the URI to be used to identify the IDMEF namespace.

1.2.2. Rationale for Implementing IDMEF in XML

XML-based applications are being used or developed for a wide variety of purposes, including electronic data interchange in a variety of fields, financial data interchange, electronic business cards, calendar and scheduling, enterprise software distribution, web "push" technology, and markup languages for chemistry, mathematics, music, molecular dynamics, astronomy, book and periodical publishing, web publishing, weather observations, real estate transactions, and many others.

XML’s flexibility makes it a good choice for these applications; that same flexibility makes it a good choice for implementing the IDMEF as well. Other, more specific reasons for choosing XML to implement the IDMEF are:

- XML allows a custom language to be developed specifically for the purpose of describing intrusion detection alerts. It also defines a standard way to extend this language, either for later revisions of this document ("standard" extensions) or for vendor-specific use ("non-standard" extensions).

- Software tools for processing XML documents are widely available, in both commercial and open source forms. Numerous tools and APIs for parsing and/or validating XML are available in a variety of languages, including Java, C, C++, Tcl, Perl, Python, and GNU Emacs Lisp. Widespread access to tools will make adoption of the IDMEF by product developers easier, and hopefully, faster.

- XML meets IDMEF Requirement 5.1 [2], that message formats support full internationalization and localization. The XML standard requires support for both the UTF-8 and UTF-16 encodings of ISO/IEC 10646 (Universal Multiple-Octet Coded Character Set, "UCS") and Unicode, making all XML applications (and therefore all IDMEF-compliant applications) compatible with these common character encodings.

  XML also provides support for specifying, on a per-element basis, the language in which the element's content is written, making IDMEF easy to adapt to "Natural Language Support" versions of a product.

- XML meets IDMEF Requirement 5.2 [2], that message formats must support filtering and aggregation. XML's integration with XSL, a style language, allows messages to be combined, discarded, and rearranged.

- Ongoing XML development projects, in the W3C and elsewhere, will provide object-oriented extensions, database support, and other useful features. If implemented in XML, the IDMEF immediately gains these features as well.

- XML is free, with no license, no license fees, and no royalties.

2. Notices and Conventions Used in This Document

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

An "IDMEF-compliant application" is a program or program component, such as an analyzer or manager, that reads and/or writes messages in the format specified by this memo.

An "IDMEF document" is a message that adheres to the requirements specified by this memo and that is exchanged by two or more IDMEF applications. "IDMEF message" is another term for an "IDMEF document".

3. Notational Conventions and Formatting Issues

This document uses three notations: Unified Modeling Language to describe the data model [14], XML to describe the markup used in IDMEF documents, and IDMEF markup to represent the documents themselves.

3.1. IDMEF XML Documents

This section describes IDMEF XML document formatting rules. Most of these rules are "inherited" from the rules for formatting XML documents.

3.1.1. The Document Prolog

The format of an IDMEF XML document prolog is described in the following sections.

3.1.1.1. XML Declaration

IDMEF documents being exchanged between IDMEF-compliant applications MUST begin with an XML declaration, and MUST specify the XML version in use. Specification of the encoding in use is RECOMMENDED.

An IDMEF message SHOULD therefore start with:

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef"/>
```

IDMEF-compliant applications MAY choose to omit the XML declaration internally to conserve space, adding it only when the message is sent to another destination (e.g., a web browser). This practice is NOT RECOMMENDED unless it can be accomplished without loss of each message's version and encoding information.

In order to be valid (see Section 6.1), an XML document must contain a document type definition. However, this represents significant overhead to an IDMEF-compliant application, both in the bandwidth it consumes as well as the requirements it places on the XML processor (not only to parse the declaration itself, but also to parse the DTD it references).

Implementors MAY decide, therefore, to have analyzers and managers agree out-of-band on the particular document type definition they will be using to exchange messages (the standard one as defined here, or one with extensions), and then omit the document type definition from IDMEF messages. The method for negotiating this agreement is outside the scope of this document. Note that great care must be taken in negotiating any such agreements, as the manager may have to accept messages from many different analyzers, each using a DTD with a different set of extensions.

3.1.2. Character Data Processing in IDMEF

For portability reasons, IDMEF-compliant applications SHOULD NOT use, and IDMEF messages SHOULD NOT be encoded in, character encodings other than UTF-8 and UTF-16. Consistent with the XML standard, if no encoding is specified for an IDMEF message, UTF-8 is assumed.

NOTE: The ASCII character set is a subset of the UTF-8 encoding, and therefore may be used to encode IDMEF messages.

Per the XML standard, IDMEF documents encoded in UTF-16 MUST begin with the Byte Order Mark described by ISO/IEC 10646 Annex E and Unicode Appendix B (the "ZERO WIDTH NO-BREAK SPACE" character, #xFEFF).

3.1.2.1. Character Entity References

It is RECOMMENDED that IDMEF-compliant applications use the entity reference form (see Section 3.2.3.1) of the characters '&', ,'<', '>', '"', and ''' (single-quote) whenever writing these characters in data, to avoid any possibility of misinterpretation.

3.1.2.2. White Space Processing

All IDMEF elements MUST support the "xml:space" attribute.

3.1.3. Languages in IDMEF

IDMEF-compliant applications MUST specify the language in which their contents are encoded; in general this can be done by specifying the "xml:lang" attribute for the top-level element and letting all other elements "inherit" that definition [10].

3.2. IDMEF Data Types

Within an XML IDMEF message, all data will be expressed as "text" (as opposed to "binary"), since XML is a text formatting language. We provide typing information for the attributes of the classes in the data model, however, to convey to the reader the type of data that the model expects for each attribute.

Each data type in the model has specific formatting requirements in an XML IDMEF message; these requirements are set forth in this section.

3.2.1. Integers

Integer attributes are represented by the INTEGER data type. Integer data MUST be encoded in Base 10 or Base 16.

Base 10 integer encoding uses the digits '0' through '9' and an optional sign ('+' or '-'). For example, "123", "-456".

Base 16 integer encoding uses the digits '0' through '9' and 'a' through 'f' (or their uppercase equivalents), and is preceded by the characters "0x". For example, "0x1a2b".

3.2.2. Real Numbers

Real (floating-point) attributes are represented by the REAL data type. Real data MUST be encoded in Base 10.

Real encoding is that of the POSIX 1003.1 "strtod" library function: an optional sign ('+' or '-') followed by a non-empty string of decimal digits, optionally containing a radix character, then an optional exponent part. An exponent part consists of an 'e' or 'E', followed by an optional sign, followed by one or more decimal digits. For example, "123.45e02", "-567,89e-03".

IDMEF-compliant applications MUST support both the '.' and ',' radix characters.

3.2.3. Characters and Strings

Single-character attributes are represented by the CHARACTER data type. Multi-character attributes of known length are represented by the STRING data type.

Character and string data have no special formatting requirements, other than the need to occasionally use character references (see Section 3.2.3.1 and Section 3.2.3.2) to represent special characters.

3.2.3.1. Character Entity References

Within XML documents, certain characters have special meanings in some contexts. To include the actual character itself in one of these contexts, a special escape sequence, called an entity reference, must be used.

The characters that sometimes need to be escaped, and their entity references, are:

Character	Entity Reference
&	&amp;
<	&lt;
>	&gt;
"	&quot;
'	&apos;

3.2.3.2. Character Code References

Any character defined by the ISO/IEC 10646 and Unicode standards may be included in an XML document by the use of a character reference. A character reference is started with the characters '&' and '#', and ended with the character ';'. Between these characters, the character code for the character is inserted.

If the character code is preceded by an 'x' it is interpreted in hexadecimal (base 16); otherwise, it is interpreted in decimal (base 10). For instance, the ampersand (&) is encoded as & or & and the less-than sign (<) is encoded as < or <.

Any one-, two-, or four-byte character specified in the ISO/IEC 10646 and Unicode standards can be included in a document using this technique.

3.2.4. Bytes

Binary data is represented by the BYTE (and BYTE[]) data type.

Binary data MUST be encoded in its entirety using base64.

3.2.5. Enumerated Types

Enumerated types are represented by the ENUM data type, and consist of an ordered list of acceptable values.

3.2.6. Date-Time Strings

Date-time strings are represented by the DATETIME data type. Each date-time string identifies a particular instant in time; ranges are not supported.

Date-time strings are formatted according to a subset of ISO 8601: 2000 [6], as show below. Section references in parentheses refer to sections of the ISO 8601:2000 standard [6].

1. Dates MUST be formatted as follows:

   YYYY-MM-DD

   where YYYY is the four-digit year, MM is the two-digit month (01-12), and DD is the two-digit day (01-31). (Section 5.2.1.1, "Complete representation -- Extended format".)

2. Times MUST be formatted as follows:

   hh:mm:ss

   where hh is the two-digit hour (00-24), mm is the two-digit minute (00-59), and ss is the two-digit second (00-60). (Section 5.3.1.1, "Complete representation -- Extended format".)

   Note that midnight has two representations, 00:00:00 and 24:00:00. Both representations MUST be supported by IDMEF-compliant applications; however, the 00:00:00 representation SHOULD be used whenever possible.

Note also that this format accounts for leap seconds. Positive leap seconds are inserted between 23:59:59Z and 24:00:00Z and are represented as 23:59:60Z. Negative leap seconds are achieved by the omission of 23:59:59Z. IDMEF-compliant applications MUST support leap seconds.

3. Times MAY be formatted to include a decimal fraction of seconds, as follows:

   hh:mm:ss.ss or
   hh:mm:ss,ss

   As many digits as necessary may follow the decimal sign (at least one digit must follow the decimal sign). Decimal fractions of hours and minutes are not supported. (Section 5.3.1.3, "Representation of decimal fractions".)

   IDMEF-compliant applications MUST support the use of both decimal signs ('.' and ',').

   Note that the number of digits in the fraction part does not imply anything about accuracy -- i.e., "00.100000", "00,1000", and "00.1" are all equivalent.

4. Times MUST be formatted to include (a) an indication that the time is in Coordinated Universal Time (UTC) or (b) an indication of the difference between the specified time and Coordinated Universal Time.

   * Times in UTC MUST be formatted by appending the letter 'Z' to the time string as follows:

     hh:mm:ssZ
     hh:mm:ss.ssZ
     hh:mm:ss,ssZ

     (Section 5.3.3, "Coordinated Universal Time (UTC) -- Extended format".)

   * If the time is ahead of or equal to UTC, a '+' sign is appended to the time string; if the time is behind UTC, a '-' sign is appended. Following the sign, the number of hours and minutes representing the different from UTC is appended, as follows:

     hh:mm:ss+hh:mm
     hh:mm:ss-hh:mm
     hh:mm:ss.ss+hh:mm

hh:mm:ss.ss-hh:mm
hh:mm:ss,ss+hh:mm
hh:mm:ss,ss-hh:mm

The difference from UTC MUST be specified in both hours and minutes, even if the minutes component is 0. A "difference" of "+00:00" is equivalent to UTC. (Section 5.3.4.2, "Local time and the difference with Coordinated Universal Time -- Extended Format".)

5. Date-time strings are created by joining the date and time strings with the letter 'T', as shown below:

YYYY-MM-DDThh:mm:ssZ
YYYY-MM-DDThh:mm:ss.ssZ
YYYY-MM-DDThh:mm:ss,ssZ
YYYY-MM-DDThh:mm:ss+hh:mm
YYYY-MM-DDThh:mm:ss-hh:mm
YYYY-MM-DDThh:mm:ss.ss+hh:mm
YYYY-MM-DDThh:mm:ss.ss-hh:mm
YYYY-MM-DDThh:mm:ss,ss+hh:mm
YYYY-MM-DDThh:mm:ss,ss-hh:mm

(Section 5.4.1, "Complete representation -- Extended format".)

In summary, IDMEF date-time strings MUST adhere to one of the nine templates identified in Paragraph 5, above.

3.2.7. NTP Timestamps

NTP timestamps are represented by the NTPSTAMP data type and are described in detail in [7] and [8]. An NTP timestamp is a 64-bit unsigned fixed-point number. The integer part is in the first 32 bits, and the fraction part is in the last 32 bits.

Within IDMEF messages, NTP timestamps MUST be encoded as two 32-bit hexadecimal values, separated by a period ('.'). For example, "0x12345678.0x87654321".

See also Section 6.4 for more information on NTP timestamps.

3.2.8. Port Lists

Port lists are represented by the PORTLIST data type and consist of a comma-separated list of numbers (individual integers) and ranges (N-M means ports N through M, inclusive). Any combination of numbers and ranges may be used in a single list. For example, "5-25,37,42,43,53,69-119,123-514".

3.2.9. Unique Identifiers

There are two types of unique identifiers used in this specification. Both types are represented by STRING data types.

These identifiers are implemented as attributes on the relevant XML elements, and they must have unique values as follows:

1. The Analyzer class' (Section 4.2.4.1) "analyzerid" attribute, if specified, MUST have a value that is unique across all analyzers in the intrusion detection environment.

   The "analyzerid" attribute is not required to be globally unique, only unique within the intrusion detection environment of which the analyzer is a member. It is permissible for two analyzers, in different intrusion detection environments, to have the same value for "analyzerid".

   The default value is "0", which indicates that the analyzer cannot generate unique identifiers.

2. The Alert and Heartbeat messages (Sections 4.2.2, 4.2.3) must be uniquely identified by the couple (analyzerid,messageid), if the analyzer supports the generation of message identifiers.

3. The Classification, Source, Target, Node, User, Process, Service, File, Address, and UserId classes' (Sections 4.2.4.2, 4.2.4.3, 4.2.4.4, 4.2.7.2, 4.2.7.3, 4.2.7.4, 4.2.7.5, 4.2.7.6, 4.2.7.2.1, and 4.2.7.3.1) "ident" attribute, if specified, MUST have a value that is unique across all messages sent by the individual analyzer.

   The "ident" attribute value MUST be unique for each particular combination of data identifying an object, not for each object. Objects may have more than one "ident" value associated with them. For example, an identification of a host by name would have one value, while an identification of that host by address would have another value, and an identification of that host by both name and address would have still another value. Furthermore, different analyzers may produce different values for the same information.

   The "ident" attribute by itself provides a unique identifier only among all the "ident" values sent by a particular analyzer. But when combined with the "analyzerid" value for the analyzer, a value that is unique across the intrusion detection environment is created. Again, there is no requirement for global uniqueness.

The default value is "0", which indicates that the analyzer cannot generate unique identifiers.

The specification of methods for creating the unique values contained in these attributes is outside the scope of this document.

4. The IDMEF Data Model and DTD

In this section, the individual components of the IDMEF data model are explained in detail. Unified Modeling Language (UML) diagrams of the model are provided to show how the components are related to each other, and relevant sections of the IDMEF DTD are presented to show how the model is translated into XML.

4.1. Data Model Overview

The relationship between the principal components of the data model is shown in Figure 1 (occurrence indicators and attributes are omitted).

The top-level class for all IDMEF messages is IDMEF-Message; each type of message is a subclass of this top-level class. There are presently two types of messages defined: Alerts and Heartbeats. Within each message, subclasses of the message class are used to provide the detailed information carried in the message.

It is important to note that the data model does not specify how an alert should be classified or identified. For example, a port scan may be identified by one analyzer as a single attack against multiple targets, while another analyzer might identify it as multiple attacks from a single source. However, once an analyzer has determined the type of alert it plans to send, the data model dictates how that alert should be formatted.

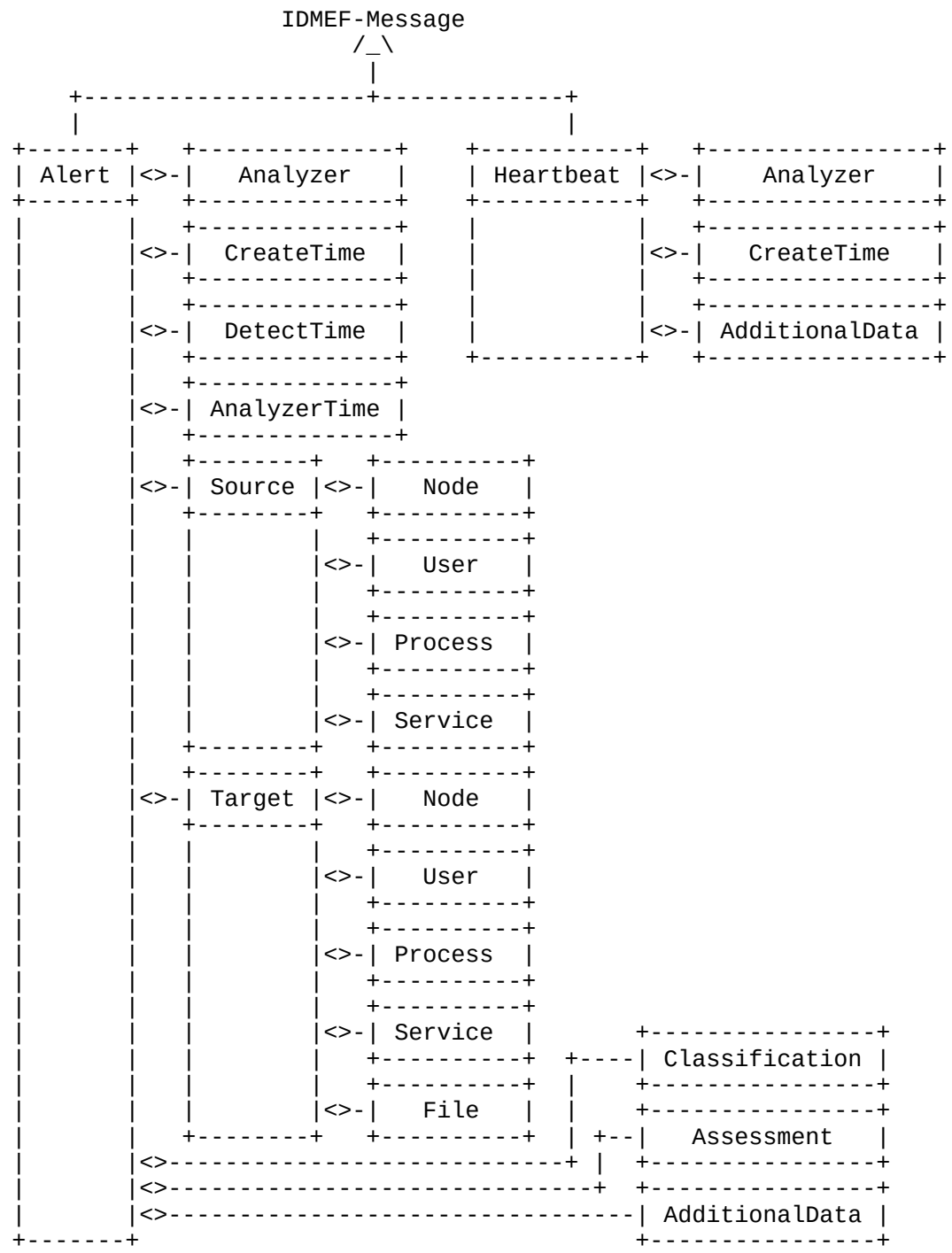
IDMEF-Message
Alert
Analyzer
CreateTime
DetectTime
AnalyzerTime
Source
Node
User
Process
Service
Target
Node
User
Process
Service
File
Heartbeat
Analyzer
CreateTime
AdditionalData
Classification
Assessment
AdditionalData

Figure 1: Data Model Overview

4.2. The Message Classes

The individual classes are described in the following sections.

4.2.1. The IDMEF-Message Class

All IDMEF messages are instances of the IDMEF-Message class; it is the top-level class of the IDMEF data model, as well as the IDMEF DTD. There are currently two types (subclasses) of IDMEF-Message: Alert and Heartbeat.

The IDMEF-Message class has a single attribute:

version

The version of the IDMEF-Message specification (this document) this message conforms to. Applications specifying a value for this attribute MUST specify the value "1.0".

4.2.2. The Alert Class

Generally, every time an analyzer detects an event that it has been configured to look for, it sends an Alert message to its manager(s). Depending on the analyzer, an Alert message may correspond to a single detected event or multiple detected events. Alerts occur asynchronously in response to outside events.

An Alert message is composed of several aggregate classes, as shown in Figure 2. The aggregate classes themselves are described in Section 4.2.4, Section 4.2.5, and Section 4.2.6.

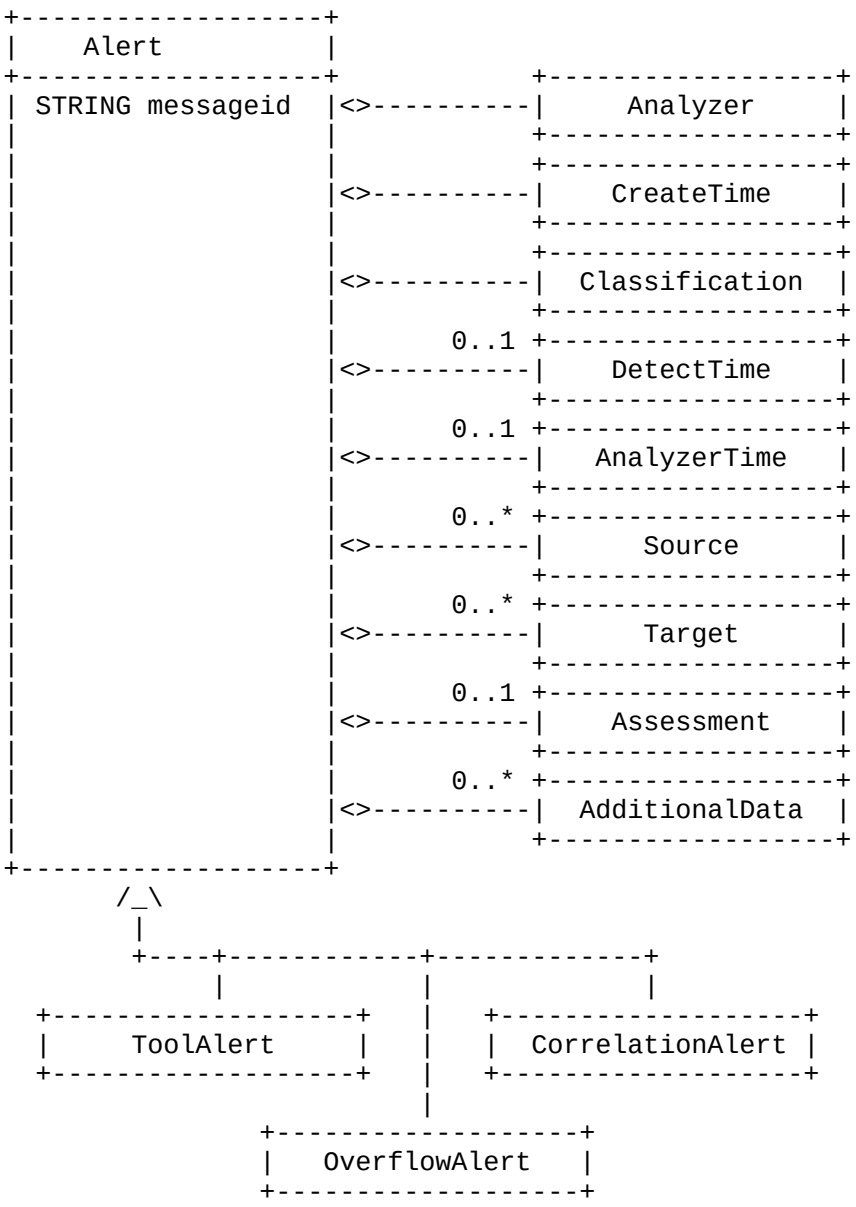

Figure 2: The Alert Class

The aggregate classes that make up Alert are:

Analyzer

Exactly one. Identification information for the analyzer that originated the alert.

CreateTime

Exactly one. The time the alert was created. Of the three times that may be provided with an Alert, this is the only one that is required.

Classification

Exactly one. The "name" of the alert, or other information allowing the manager to determine what it is.

DetectTime

Zero or one. The time the event(s) leading up to the alert was detected. In the case of more than one event, the time the first event was detected. In some circumstances, this may not be the same value as CreateTime.

AnalyzerTime

Zero or one. The current time on the analyzer (see Section 6.3).

Source

Zero or more. The source(s) of the event(s) leading up to the alert.

Target

Zero or more. The target(s) of the event(s) leading up to the alert.

Assessment

Zero or one. Information about the impact of the event, actions taken by the analyzer in response to it, and the analyzer's confidence in its evaluation.

AdditionalData

Zero or more. Information included by the analyzer that does not fit into the data model. This may be an atomic piece of data, or a large amount of data provided through an extension to the IDMEF (see Section 5).

Alert is represented in the IDMEF DTD as follows:

```
<!ELEMENT Alert                          (
    Analyzer, CreateTime, DetectTime?, AnalyzerTime?,
    Source*, Target*, Classification, Assessment?, (ToolAlert |
    OverflowAlert | CorrelationAlert)?, AdditionalData*
  )>
<!ATTLIST Alert
    messageid           CDATA                  '0'
    %attlist.global;
  >
```

The Alert class has one attribute:

messageid

Optional. A unique identifier for the alert; see Section 3.2.9.

4.2.2.1. The ToolAlert Class

The ToolAlert class carries additional information related to the use of attack tools or malevolent programs such as Trojan horses and can be used by the analyzer when it is able to identify these tools. It is intended to group one or more previously-sent alerts together, to say "these alerts were all the result of someone using this tool".

The ToolAlert class is composed of three aggregate classes, as shown in Figure 3.

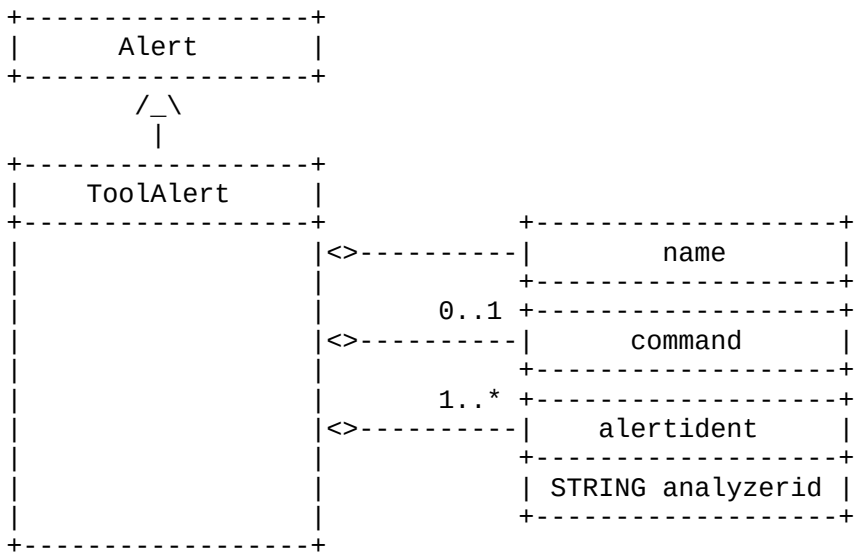

Figure 3: The ToolAlert Class

The aggregate classes that make up ToolAlert are:

name

Exactly one. STRING. The reason for grouping the alerts together, for example, the name of a particular tool.

command

Zero or one. STRING. The command or operation that the tool was asked to perform, for example, a BackOrifice ping.

alertident

One or more. STRING. The list of alert identifiers that are related to this alert. Because alert identifiers are only unique across the alerts sent by a single analyzer, the optional "analyzerid" attribute of "alertident" should be used to identify the analyzer that a particular alert came from. If the "analyzerid" is not provided, the alert is assumed to have come from the same analyzer that is sending the ToolAlert.

This is represented in the IDMEF DTD as follows:

```
<!ELEMENT ToolAlert                    (
    name, command?, alertident+
  )>
<!ATTLIST ToolAlert
    %attlist.global;
  >
```

4.2.2.2. The CorrelationAlert Class

The CorrelationAlert class carries additional information related to the correlation of alert information. It is intended to group one or more previously-sent alerts together, to say "these alerts are all related".

The CorrelationAlert class is composed of two aggregate classes, as shown in Figure 4.

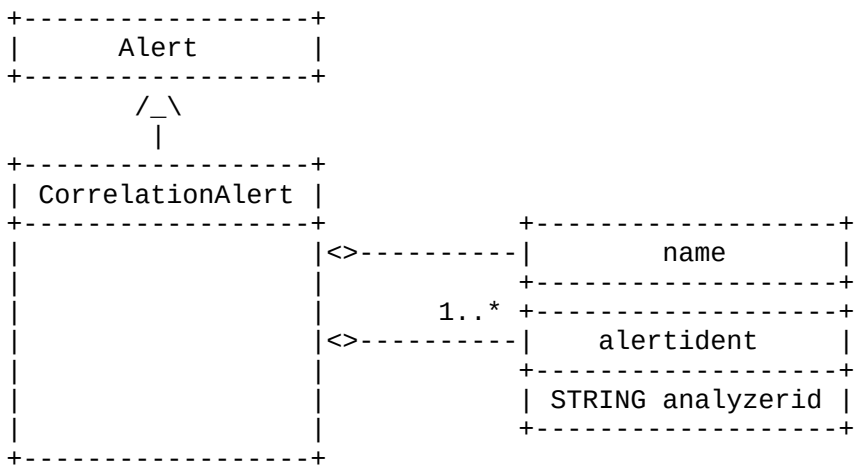

Figure 4: The CorrelationAlert Class

The aggregate classes that make up CorrelationAlert are:

name

Exactly one. STRING. The reason for grouping the alerts together, for example, a particular correlation method.

alertident

One or more. STRING. The list of alert identifiers that are related to this alert. Because alert identifiers are only unique across the alerts sent by a single analyzer, the optional "analyzerid" attribute of "alertident" should be used to identify the analyzer that a particular alert came from. If the "analyzerid" is not provided, the alert is assumed to have come from the same analyzer that is sending the CorrelationAlert.

This is represented in the IDMEF DTD as follows.

```
<!ELEMENT CorrelationAlert             (
    name, alertident+
  )>
<!ATTLIST CorrelationAlert
    %attlist.global;
  >
```

4.2.2.3. The OverflowAlert Class

The OverflowAlert carries additional information related to buffer overflow attacks. It is intended to enable an analyzer to provide the details of the overflow attack itself.

The OverflowAlert class is composed of three aggregate classes, as shown in Figure 5.

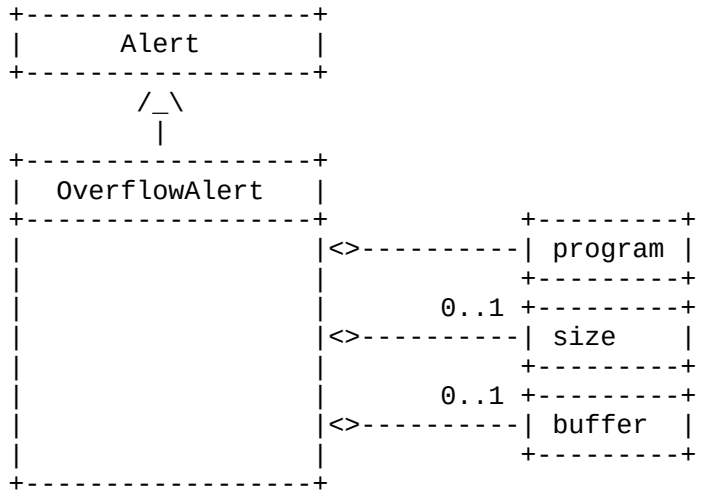

Figure 5: The OverflowAlert Class

The aggregate classes that make up OverflowAlert are:

program

Exactly one. STRING. The program that the overflow attack attempted to run (NOTE: this is not the program that was attacked).

size

Zero or one. INTEGER. The size, in bytes, of the overflow (i.e., the number of bytes the attacker sent).

buffer

Zero or one. BYTE[]. Some or all of the overflow data itself (dependent on how much the analyzer can capture).

This is represented in the IDMEF DTD as follows:

```
<!ELEMENT OverflowAlert                (
    program, size?, buffer?
  )>
<!ATTLIST OverflowAlert
    %attlist.global;
  >
```

4.2.3. The Heartbeat Class

Analyzers use Heartbeat messages to indicate their current status to managers. Heartbeats are intended to be sent in a regular period, say, every ten minutes or every hour. The receipt of a Heartbeat message from an analyzer indicates to the manager that the analyzer is up and running; lack of a Heartbeat message (or more likely, lack of some number of consecutive Heartbeat messages) indicates that the analyzer or its network connection has failed.

All managers MUST support the receipt of Heartbeat messages; however, the use of these messages by analyzers is OPTIONAL. Developers of manager software SHOULD permit the software to be configured on a per-analyzer basis to use/not use Heartbeat messages.

A Heartbeat message is composed of several aggregate classes, as shown in Figure 6. The aggregate classes themselves are described in Sections 4.2.4 and 4.2.5.

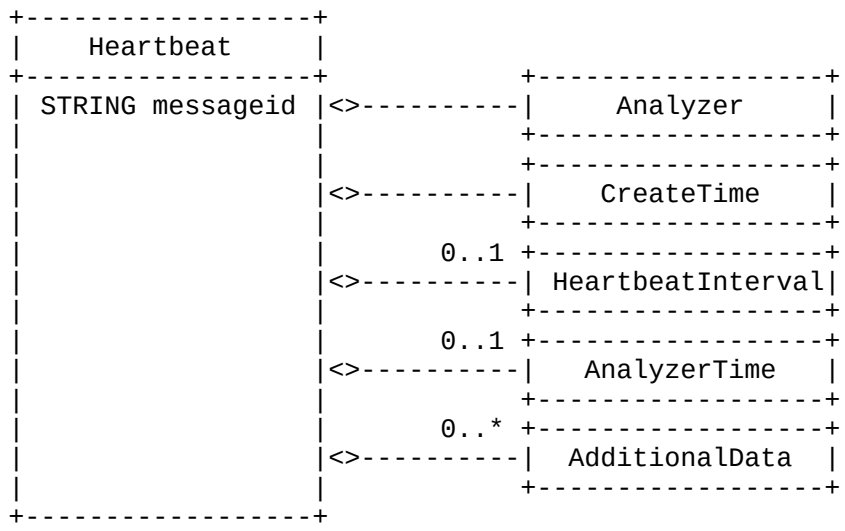

Figure 6: The Heartbeat Class

The aggregate classes that make up Heartbeat are:

Analyzer

Exactly one. Identification information for the analyzer that originated the heartbeat.

CreateTime

Exactly one. The time the heartbeat was created.

HeartbeatInterval

Zero or one. The interval in seconds at which heartbeats are generated.

AnalyzerTime

Zero or one. The current time on the analyzer (see Section 6.3).

AdditionalData

Zero or more. Information included by the analyzer that does not fit into the data model. This may be an atomic piece of data or a large amount of data provided through an extension to the IDMEF (see Section 5).

This is represented in the IDMEF DTD as follows:

```
<!ELEMENT Heartbeat                     (
    Analyzer, CreateTime, HeartbeatInterval?, AnalyzerTime?,
    AdditionalData*
  )>
<!ATTLIST Heartbeat
    messageid           CDATA                   '0'
    %attlist.global;
  >
```

The Heartbeat class has one attribute:

messageid

Optional. A unique identifier for the heartbeat; see Section 3.2.9.

4.2.4. The Core Classes

The core classes -- Analyzer, Source, Target, Classification, and AdditionalData -- are the main parts of Alerts and Heartbeats, as shown in Figure 7.

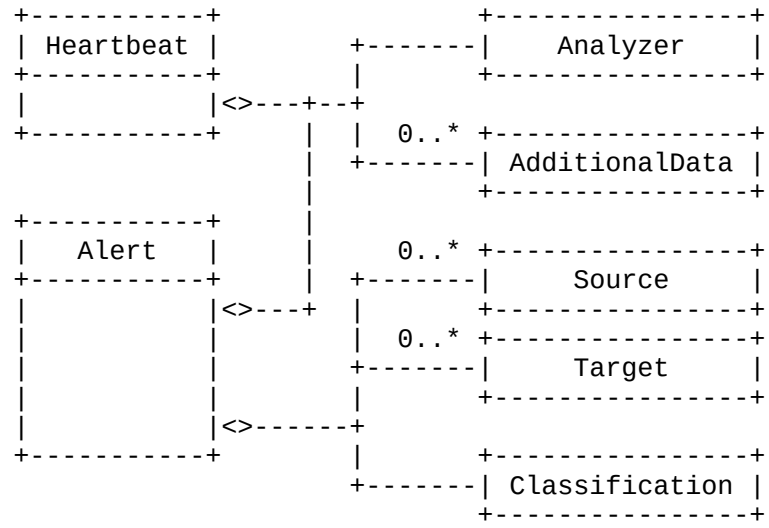

Figure 7: The Core Classes

4.2.4.1. The Analyzer Class

The Analyzer class identifies the analyzer from which the Alert or Heartbeat message originates. Only one analyzer may be encoded for each alert or heartbeat, and that MUST be the analyzer at which the alert or heartbeat originated. Although the IDMEF data model does not prevent the use of hierarchical intrusion detection systems (where alerts get relayed up the tree), it does not provide any way to record the identity of the "relay" analyzers along the path from the originating analyzer to the manager that ultimately receives the alert.

The Analyzer class is composed of three aggregate classes, as shown in Figure 8.

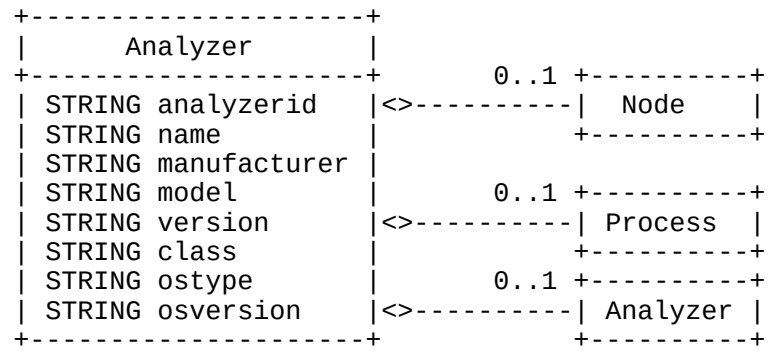

Figure 8: The Analyzer Class

The aggregate classes that make up Analyzer are:

Node

Zero or one. Information about the host or device on which the analyzer resides (network address, network name, etc.).

Process

Zero or one. Information about the process in which the analyzer is executing.

Analyzer

Zero or one. Information about the analyzer from which the message may have gone through. The idea behind this mechanism is that when a manager receives an alert and wants to forward it to another analyzer, it needs to substitute the original analyzer

information with its own. To preserve the original analyzer information, it may be included in the new analyzer definition. This will allow analyzer path tracking.

This is represented in the IDMEF DTD as follows:

```
<!ELEMENT Analyzer                      (
    Node?, Process?, Analyzer?
  )>
<!ATTLIST Analyzer
    analyzerid          CDATA                   '0'
    name                CDATA                   #IMPLIED
    manufacturer        CDATA                   #IMPLIED
    model               CDATA                   #IMPLIED
    version             CDATA                   #IMPLIED
    class               CDATA                   #IMPLIED
    ostype              CDATA                   #IMPLIED
    osversion           CDATA                   #IMPLIED
    %attlist.global;
  >
```

The Analyzer class has eight attributes:

analyzerid

Optional (but see below). A unique identifier for the analyzer; see Section 3.2.9.

This attribute is only "partially" optional. If the analyzer makes use of the "ident" attributes on other classes to provide unique identifiers for those objects, then it MUST also provide a valid "analyzerid" attribute. This requirement is dictated by the uniqueness requirements of the "ident" attribute (they are unique only within the context of a particular "analyzerid"). If the analyzer does not make use of the "ident" attributes, however, it may also omit the "analyzerid" attribute.

name

Optional. An explicit name for the analyzer that may be easier to understand than the analyzerid.

manufacturer

Optional. The manufacturer of the analyzer software and/or hardware.

model

Optional. The model name/number of the analyzer software and/or hardware.

version

Optional. The version number of the analyzer software and/or hardware.

class

Optional. The class of analyzer software and/or hardware.

ostype

Optional. Operating system name. On POSIX 1003.1 compliant systems, this is the value returned in utsname.sysname by the uname() system call, or the output of the "uname -s" command.

osversion

Optional. Operating system version. On POSIX 1003.1 compliant systems, this is the value returned in utsname.release by the uname() system call, or the output of the "uname -r" command.

The "manufacturer", "model", "version", and "class" attributes' contents are vendor-specific, but may be used together to identify different types of analyzers (and perhaps make determinations about the contents to expect in other vendor-specific fields of IDMEF messages).

4.2.4.2. The Classification Class

The Classification class provides the "name" of an alert, or other information allowing the manager to determine what it is. This name is chosen by the alert provider.

The Classification class is composed of one aggregate class, as shown in Figure 9.

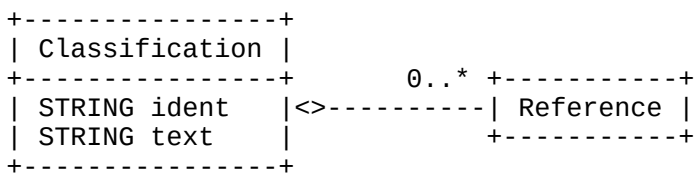

Figure 9: The Classification Class

The aggregate class that makes up Classification is:

Reference

Zero or more. Information about the message, pointing to external documentation sites, that will provide background information about the alert.

This is represented in the IDMEF DTD as follows:

```
<!ELEMENT Classification               (
    Reference*
  )>
<!ATTLIST Classification
    ident               CDATA                  '0'
    text                CDATA                  #REQUIRED
  >
```

The Classification class has two attributes:

ident

Optional. A unique identifier for this classification; see Section 3.2.9.

text

Required. A vendor-provided string identifying the Alert message.

4.2.4.3. The Source Class

The Source class contains information about the possible source(s) of the event(s) that generated an alert. An event may have more than one source (e.g., in a distributed denial-of-service attack).

The Source class is composed of four aggregate classes, as shown in Figure 10.

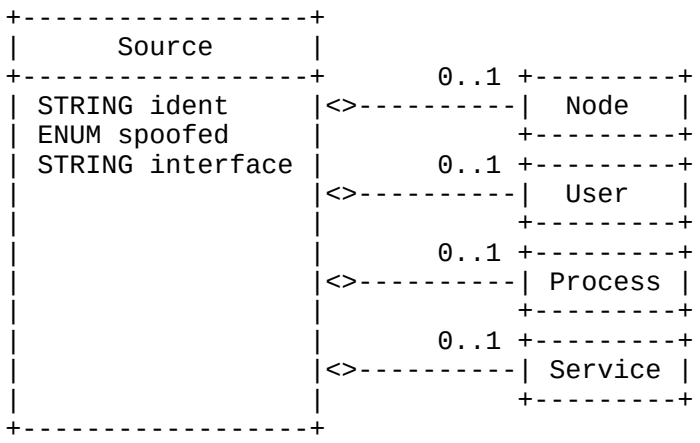

Figure 10: The Source Class

The aggregate classes that make up Source are:

Node

Zero or one. Information about the host or device that appears to be causing the events (network address, network name, etc.).

User

Zero or one. Information about the user that appears to be causing the event(s).

Process

Zero or one. Information about the process that appears to be causing the event(s).

Service

Zero or one. Information about the network service involved in the event(s).

This is represented in the IDMEF DTD as follows:

```
<!ELEMENT Source                          (
    Node?, User?, Process?, Service?
  )>
<!ATTLIST Source
    ident               CDATA                   '0'
    spoofed             %attvals.yesno;         'unknown'
    interface           CDATA                   #IMPLIED
    %attlist.global;
  >
```

The Source class has three attributes:

ident

Optional. A unique identifier for this source; see Section 3.2.9.

spoofed

Optional. An indication of whether the source is, as far as the analyzer can determine, a spoofed address used for hiding the real origin of the attack. The permitted values for this attribute are shown below. The default value is "unknown". (See also Section 10.)

Rank	Keyword	Description
0	unknown	Accuracy of source information unknown
1	yes	Source is believed to be a decoy
2	no	Source is believed to be "real"

interface

Optional. May be used by a network-based analyzer with multiple interfaces to indicate which interface this source was seen on.

4.2.4.4. The Target Class

The Target class contains information about the possible target(s) of the event(s) that generated an alert. An event may have more than one target (e.g., in the case of a port sweep).

The Target class is composed of four aggregate classes, as shown in Figure 11.

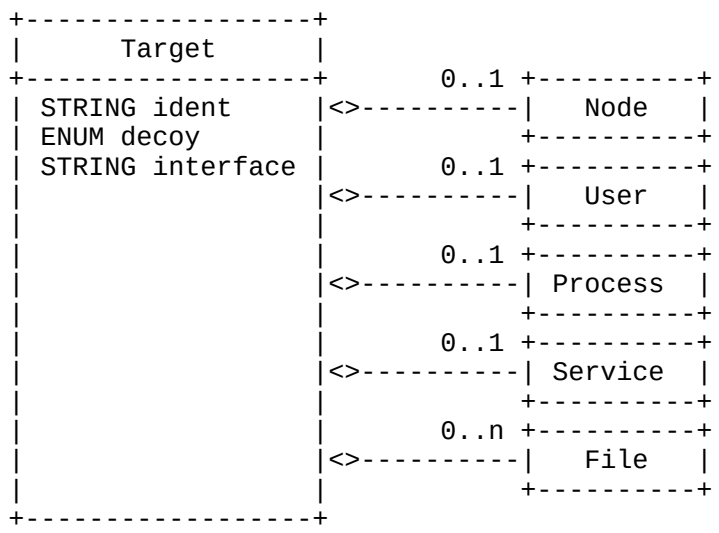

Figure 11: The Target Class

The aggregate classes that make up Target are:

Node

Zero or one. Information about the host or device at which the event(s) (network address, network name, etc.) is being directed.

User

Zero or one. Information about the user at which the event(s) is being directed.

Process

Zero or one. Information about the process at which the event(s) is being directed.

Service

Zero or one. Information about the network service involved in the event(s).

File

Optional. Information about file(s) involved in the event(s).

This is represented in the IDMEF DTD as follows:

```
<!ELEMENT Target                        (
    Node?, User?, Process?, Service?, File*
  )>
<!ATTLIST Target
    ident               CDATA                   '0'
    decoy               %attvals.yesno;         'unknown'
    interface           CDATA                   #IMPLIED
    %attlist.global;
  >
```

The Target class has three attributes:

ident

Optional. A unique identifier for this target, see Section 3.2.9.

decoy

Optional. An indication of whether the target is, as far as the analyzer can determine, a decoy. The permitted values for this attribute are shown below. The default value is "unknown". (See also Section 10.)

Rank	Keyword	Description
0	unknown	Accuracy of target information unknown
1	yes	Target is believed to be a decoy
2	no	Target is believed to be "real"

interface

Optional. May be used by a network-based analyzer with multiple interfaces to indicate which interface this target was seen on.

4.2.4.5. The Assessment Class

The Assessment class is used to provide the analyzer's assessment of an event -- its impact, actions taken in response, and confidence.

The Assessment class is composed of three aggregate classes, as shown in Figure 12.

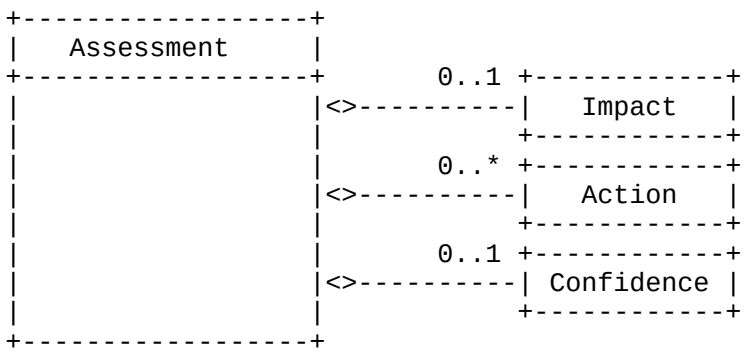

Figure 12: The Assessment Class

The aggregate classes that make up Assessment are:

Impact

Zero or one. The analyzer's assessment of the impact of the event on the target(s).

Action

Zero or more. The action(s) taken by the analyzer in response to the event.

Confidence

Zero or one. A measurement of the confidence the analyzer has in its evaluation of the event.

This is represented in the IDMEF DTD as follows:

```
<!ELEMENT Assessment                    (
    Impact?, Action*, Confidence?
  )>
<!ATTLIST Assessment
    %attlist.global;
  >
```

4.2.4.6. The AdditionalData Class

The AdditionalData class is used to provide information that cannot be represented by the data model. AdditionalData can be used to provide atomic data (integers, strings, etc.) in cases where only small amounts of additional information need to be sent; it can also be used to extend the data model and the DTD to support the transmission of complex data (such as packet headers). Detailed instructions for extending the data model and the DTD are provided in Section 5.

Rank	Keyword	Description
0	boolean	The element contains a boolean value, i.e., the strings "true" or "false"
1	byte	The element content is a single 8-bit byte (see Section 3.2.4)
2	character	The element content is a single character (see Section 3.2.3)
3	date-time	The element content is a date-time string (see Section 3.2.6)
4	integer	The element content is an integer (see Section 3.2.1)
5	ntpstamp	The element content is an NTP timestamp (see Section 3.2.7)
6	portlist	The element content is a list of ports (see Section 3.2.8)
7	real	The element content is a real number (see Section 3.2.2)
8	string	The element content is a string (see Section 3.2.3)
9	byte-string	The element is a byte[] (see Section 3.2.4)
10	xmltext	The element content is XML-tagged data (see Section 5.2)

The AdditionalData element is declared in the IDMEF DTD as follows:

```
<!ENTITY % attvals.adtype               "
    ( boolean | byte | character | date-time | integer | ntpstamp |
      portlist | real | string | byte-string | xmltext )
  ">

<!ELEMENT AdditionalData          (
  (boolean | byte        | character | date-time |
   integer | ntpstamp    | portlist  | real      |
   string  | byte-string | xmltext  )
 )>

<!ATTLIST AdditionalData
    type               %attvals.adtype;        'string'
    meaning            CDATA                   #IMPLIED
    %attlist.global;
  >
```

The AdditionalData class has one attribute:

meaning

Optional. A string describing the meaning of the element content. These values will be vendor/implementation dependent; the method for ensuring that managers understand the strings sent by analyzers is outside the scope of this specification. A list of acceptable meaning keywords is not within the scope of the document, although later versions may undertake to establish such a list.

4.2.5. The Time Classes

The data model provides three classes for representing time. These classes are elements of the Alert and Heartbeat classes.

The time classes are represented in the IDMEF DTD as follows:

```
<!ELEMENT ntpstamp          (#PCDATA)         >
<!ATTLIST ntpstamp          %attlist.global;  >

<!ELEMENT CreateTime          (#PCDATA) >
<!ATTLIST CreateTime
    ntpstamp            CDATA                   #REQUIRED
    %attlist.global;
  >

<!ELEMENT DetectTime          (#PCDATA) >
<!ATTLIST DetectTime
    ntpstamp            CDATA                   #REQUIRED
    %attlist.global;
  >

<!ELEMENT AnalyzerTime        (#PCDATA) >
<!ATTLIST AnalyzerTime
    ntpstamp            CDATA                   #REQUIRED
    %attlist.global;
  >
```

The DATETIME format of the <CreateTime> element content is described in Section 3.2.6.

If the date and time represented by the element content and the NTP timestamp differ (should "never" happen), the value in the NTP timestamp MUST be used.

4.2.5.1. The CreateTime Class

The CreateTime class is used to indicate the date and time the alert or heartbeat was created by the analyzer.

4.2.5.2. The DetectTime Class

The DetectTime class is used to indicate the date and time that the event(s) producing an alert was detected by the analyzer. In the case of more than one event, it is the time that the first event was detected. (This may or may not be the same time as CreateTime; analyzers are not required to send alerts immediately upon detection).

4.2.5.3. The AnalyzerTime Class

The AnalyzerTime class is used to indicate the current date and time on the analyzer. Its values should be filled in as late as possible in the message transmission process, ideally immediately before placing the message "on the wire".

The use of <AnalyzerTime> to perform rudimentary time synchronization between analyzers and managers is discussed in Section 6.3.

4.2.6. The Assessment Classes

The data model provides three types of "assessments" that an analyzer can make about an event. These classes are aggregates of the Assessment class.

4.2.6.1. The Impact Class

The Impact class is used to provide the analyzer's assessment of the impact of the event on the target(s). It is represented in the IDMEF DTD as follows:

```
<!ENTITY % attvals.severity             "
    ( info | low | medium | high )
  ">
<!ENTITY % attvals.completion           "
    ( failed | succeeded )
  ">
<!ENTITY % attvals.impacttype           "
    ( admin | dos | file | recon | user | other )
  ">

<!ELEMENT Impact              (#PCDATA) >
<!ATTLIST Impact
    severity           %attvals.severity;      #IMPLIED
    completion         %attvals.completion;    #IMPLIED
    type               %attvals.impacttype;    'other'
    %attlist.global;
  >
```

The Impact class has three attributes:

severity

An estimate of the relative severity of the event. The permitted values are shown below. There is no default value. (See also Section 10.)

Rank	Keyword	Description
0	info	Alert represents informational activity
1	low	Low severity
2	medium	Medium severity
3	high	High severity

completion

An indication of whether the analyzer believes the attempt that the event describes was successful or not. The permitted values are shown below. There is no default value. (See also Section 10.)

Rank	Keyword	Description
0	failed	The attempt was not successful
1	succeeded	The attempt succeeded

type

The type of attempt represented by this event, in relatively broad categories. The permitted values are shown below. The default value is "other". (See also Section 10.)

Rank	Keyword	Description
0	admin	Administrative privileges were attempted or obtained
1	dos	A denial of service was attempted or completed
2	file	An action on a file was attempted or completed
3	recon	A reconnaissance probe was attempted or completed
4	user	User privileges were attempted or obtained
5	other	Anything not in one of the above categories

All three attributes are optional. The element itself may be empty, or may contain a textual description of the impact, if the analyzer is able to provide additional details.

4.2.6.2. The Action Class

The Action class is used to describe any actions taken by the analyzer in response to the event. Is is represented in the IDMEF DTD as follows:

```
<!ENTITY % attvals.actioncat            "
    ( block-installed | notification-sent | taken-offline | other )
  ">

<!ELEMENT Action              (#PCDATA) >
<!ATTLIST Action
    category           %attvals.actioncat;     'other'
    %attlist.global;
  >
```

Action has one attribute:

category

The type of action taken. The permitted values are shown below. The default value is "other". (See also Section 10.)

Rank	Keyword	Description
0	block-installed	A block of some sort was installed to prevent an attack from reaching its destination. The block could be a port block, address block, etc., or disabling a user account.
1	notification-sent	A notification message of some sort was sent out-of-band (via pager, e-mail, etc.). Does not include the transmission of this alert.
2	taken-offline	A system, computer, or user was taken offline, as when the computer is shut down or a user is logged off.
3	other	Anything not in one of the above categories.

The element itself may be empty, or may contain a textual description of the action, if the analyzer is able to provide additional details.

4.2.6.3. The Confidence Class

The Confidence class is used to represent the analyzer's best estimate of the validity of its analysis. It is represented in the IDMEF DTD as follows:

```
<!ENTITY % attvals.rating               "
    ( low | medium | high | numeric )
  ">

<!ELEMENT Confidence          (#PCDATA) >
<!ATTLIST Confidence
    rating              %attvals.rating;        'numeric'
    %attlist.global;
  >
```

The Confidence class has one attribute:

rating

The analyzer's rating of its analytical validity. The permitted values are shown below. The default value is "numeric". (See also Section 10.)

Rank	Keyword	Description
0	low	The analyzer has little confidence in its validity
1	medium	The analyzer has average confidence in its validity
2	high	The analyzer has high confidence in its validity
3	numeric	The analyzer has provided a posterior probability value indicating its confidence in its validity

This element should be used only when the analyzer can produce meaningful information. Systems that can output only a rough heuristic should use "low", "medium", or "high" as the rating value. In this case, the element content should be omitted.

Systems capable of producing reasonable probability estimates should use "numeric" as the rating value and include a numeric confidence value in the element content. This numeric value should reflect a posterior probability (the probability that an attack has occurred given the data seen by the detection system and the model used by the system). It is a floating point number between 0.0 and 1.0, inclusive. The number of digits should be limited to those representable by a single precision floating point value, and may be represented as described in Section 3.2.2.

NOTE: It should be noted that different types of analyzers may compute confidence values in different ways and that in many cases, confidence values from different analyzers should not be compared (for example, if the analyzers use different methods of computing or representing confidence, or are of different types or configurations). Care should be taken when implementing systems that process confidence values (such as event correlators) not to make comparisons or assumptions that cannot be supported by the system's knowledge of the environment in which it is working.

4.2.7. The Support Classes

The support classes make up the major parts of the core classes, and are shared between them.

4.2.7.1. The Reference Class

The Reference class provides the "name" of an alert, or other information allowing the manager to determine what it is.

The Reference class is composed of two aggregate classes, as shown in Figure 13.

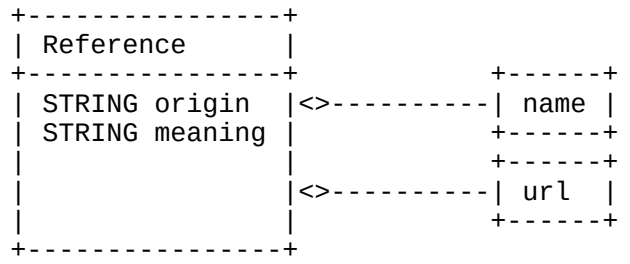

Figure 13: The Reference Class

The aggregate classes that make up Reference are:

name

Exactly one. STRING. The name of the alert, from one of the origins listed below.

url

Exactly one. STRING. A URL at which the manager (or the human operator of the manager) can find additional information about the alert. The document pointed to by the URL may include an in-depth description of the attack, appropriate countermeasures, or other information deemed relevant by the vendor.

This is represented in the IDMEF DTD as follows:

```
<!ENTITY % attvals.origin               "
    ( unknown | vendor-specific | user-specific | bugtraqid | cve |
      osvdb )
  ">

<!ELEMENT Reference               (
    name, url
  )>
<!ATTLIST Reference
    origin              %attvals.origin;        'unknown'
    meaning             CDATA                   #IMPLIED
  >
```

The Reference class has two attributes:

origin

Required. The source from which the name of the alert originates. The permitted values for this attribute are shown below. The default value is "unknown". (See also Section 10.)

Rank	Keyword	Description
0	unknown	Origin of the name is not known
1	vendor-specific	A vendor-specific name (and hence, URL); this can be used to provide product-specific information
2	user-specific	A user-specific name (and hence, URL); this can be used to provide installation-specific information
3	bugtraqid	The SecurityFocus ("Bugtraq") vulnerability database identifier (http://www.securityfocus.com/bid)
4	cve	The Common Vulnerabilities and Exposures (CVE) name (http://www.cve.mitre.org/)
5	osvdb	The Open Source Vulnerability Database (http://www.osvdb.org)

meaning

Optional. The meaning of the reference, as understood by the alert provider. This field is only valid if the value of the <origin> attribute is set to "vendor-specific" or "user-specific".

4.2.7.2. The Node Class

The Node class is used to identify hosts and other network devices (routers, switches, etc.).

The Node class is composed of three aggregate classes, as shown in Figure 14.

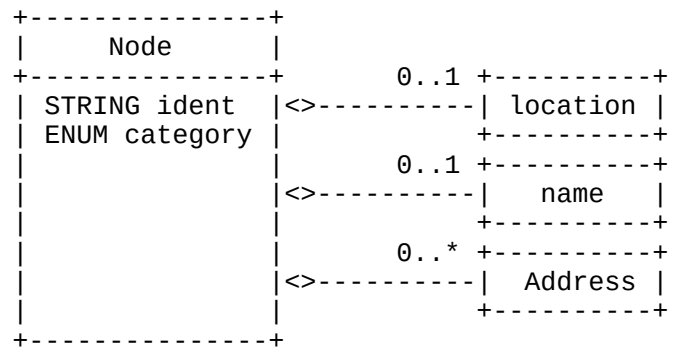

Figure 14: The Node Class

The aggregate classes that make up Node are:

location

Zero or one. STRING. The location of the equipment.

name

Zero or one. STRING. The name of the equipment. This information MUST be provided if no Address information is given.

Address

Zero or more. The network or hardware address of the equipment. Unless a name (above) is provided, at least one address must be specified.

This is represented in the IDMEF DTD as follows:

```
<!ENTITY % attvals.nodecat              "
    ( unknown | ads | afs | coda | dfs | dns | hosts | kerberos |
      nds | nis | nisplus | nt | wfw )
  ">

<!ELEMENT Node                          (
    location?, (name | Address), Address*
  )>
<!ATTLIST Node
    ident               CDATA                   '0'
    category            %attvals.nodecat;       'unknown'
    %attlist.global;
  >
```

The Node class has two attributes:

ident

Optional. A unique identifier for the node; see Section 3.2.9.

category

Optional. The "domain" from which the name information was obtained, if relevant. The permitted values for this attribute are shown in the table below. The default value is "unknown". (See also Section 10 for extensions to the table.)

Rank	Keyword	Description
0	unknown	Domain unknown or not relevant
1	ads	Windows 2000 Advanced Directory Services
2	afs	Andrew File System (Transarc)
3	coda	Coda Distributed File System
4	dfs	Distributed File System (IBM)
5	dns	Domain Name System
6	hosts	Local hosts file
7	kerberos	Kerberos realm
8	nds	Novell Directory Services
9	nis	Network Information Services (Sun)
10	nisplus	Network Information Services Plus (Sun)
11	nt	Windows NT domain
12	wfw	Windows for Workgroups

4.2.7.2.1. The Address Class

The Address class is used to represent network, hardware, and application addresses.

The Address class is composed of two aggregate classes, as shown in Figure 15.

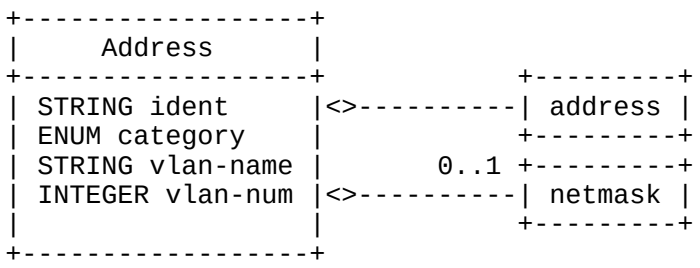

Figure 15: The Address Class

The aggregate classes that make up Address are:

address

Exactly one. STRING. The address information. The format of this data is governed by the category attribute.

netmask

Zero or one. STRING. The network mask for the address, if appropriate.

This is represented in the IDMEF DTD as follows:

```
<!ENTITY % attvals.addrcat              "
    ( unknown | atm | e-mail | lotus-notes | mac | sna | vm |
      ipv4-addr | ipv4-addr-hex | ipv4-net | ipv4-net-mask |
      ipv6-addr | ipv6-addr-hex | ipv6-net | ipv6-net-mask )
  ">

<!ELEMENT Address                       (
    address, netmask?
  )>
<!ATTLIST Address
    ident               CDATA                   '0'
    category            %attvals.addrcat;       'unknown'
    vlan-name           CDATA                   #IMPLIED
    vlan-num            CDATA                   #IMPLIED
    %attlist.global;
  >
```

The Address class has four attributes:

ident

Optional. A unique identifier for the address; see Section 3.2.9.

category

Optional. The type of address represented. The permitted values for this attribute are shown below. The default value is "unknown". (See also Section 10.)

Rank	Keyword	Description
0	unknown	Address type unknown
1	atm	Asynchronous Transfer Mode network address
2	e-mail	Electronic mail address (RFC 2822 [12])
3	lotus-notes	Lotus Notes e-mail address
4	mac	Media Access Control (MAC) address
5	sna	IBM Shared Network Architecture (SNA) address
6	vm	IBM VM ("PROFS") e-mail address
7	ipv4-addr	IPv4 host address in dotted-decimal notation (a.b.c.d)
8	ipv4-addr-hex	IPv4 host address in hexadecimal notation
9	ipv4-net	IPv4 network address in dotted-decimal notation, slash, significant bits (a.b.c.d/nn)
10	ipv4-net-mask	IPv4 network address in dotted-decimal notation, slash, network mask in dotted-decimal notation (a.b.c.d/w.x.y.z)
11	ipv6-addr	IPv6 host address
12	ipv6-addr-hex	IPv6 host address in hexadecimal notation
13	ipv6-net	IPv6 network address, slash, significant bits
14	ipv6-net-mask	IPv6 network address, slash, network mask

vlan-name

Optional. The name of the Virtual LAN to which the address belongs.

vlan-num

Optional. The number of the Virtual LAN to which the address belongs.

4.2.7.3. The User Class

The User class is used to describe users. It is primarily used as a "container" class for the UserId aggregate class, as shown in Figure 16.

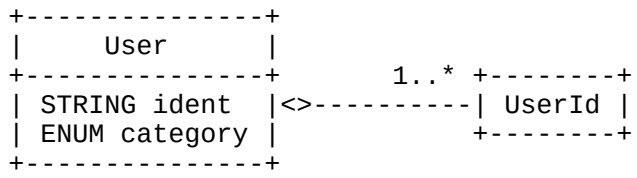

Figure 16: The User Class

The aggregate class contained in User is:

UserId

One or more. Identification of a user, as indicated by its type attribute (see Section 4.2.7.3.1).

This is represented in the IDMEF DTD as follows:

```
<!ENTITY % attvals.usercat              "
    ( unknown | application | os-device )
  ">

<!ELEMENT User                          (
    UserId+
  )>
<!ATTLIST User
    ident               CDATA                   '0'
    category            %attvals.usercat;       'unknown'
    %attlist.global;
  >
```

The User class has two attributes:

ident

Optional. A unique identifier for the user; see Section 3.2.9.

category

Optional. The type of user represented. The permitted values for this attribute are shown below. The default value is "unknown". (See also Section 10.)

Rank	Keyword	Description
0	unknown	User type unknown
1	application	An application user
2	os-device	An operating system or device user

4.2.7.3.1. The UserId Class

The UserId class provides specific information about a user. More than one UserId can be used within the User class to indicate attempts to transition from one user to another, or to provide complete information about a user's (or process') privileges.

The UserId class is composed of two aggregate classes, as shown in Figure 17.

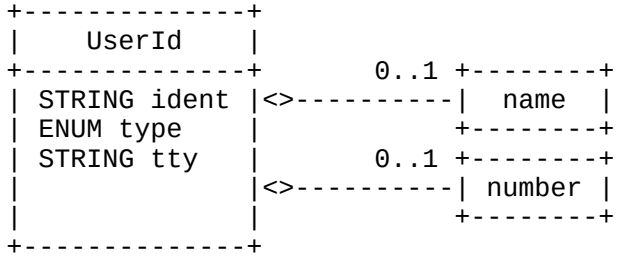

Figure 17: The UserId Class

The aggregate classes that make up UserId are:

name

Zero or one. STRING. A user or group name.

number

Zero or one. INTEGER. A user or group number.

This is represented in the IDMEF DTD as follows:

```
<!ENTITY % attvals.idtype               "
    ( current-user | original-user | target-user | user-privs |
      current-group | group-privs | other-privs )
  ">

<!ELEMENT UserId                        (
    (name, number?) | (number, name?)
  )>
<!ATTLIST UserId
    ident               CDATA                   '0'
    type                %attvals.idtype;        'original-user'
    tty                 CDATA                   #IMPLIED
    %attlist.global;
  >
```

The UserId class has three attributes:

ident

Optional. A unique identifier for the user id, see Section 3.2.9.

type

Optional. The type of user information represented. The permitted values for this attribute are shown below. The default value is "original-user". (See also Section 10.)

Rank	Keyword	Description
0	current-user	The current user id being used by the user or process. On Unix systems, this would be the "real" user id, in general.
1	original-user	The actual identity of the user or process being reported on. On those systems that (a) do some type of auditing and (b) support extracting a user id from the "audit id" token, that value should be used. On those systems that do not support this, and where the user has logged into the system, the "login id" should be used.
2	target-user	The user id the user or process is attempting to become. This would apply, on Unix systems for example, when the user attempts to use "su", "rlogin", "telnet", etc.
3	user-privs	Another user id the user or process has the ability to use, or a user id associated with a file permission. On Unix systems, this would be the "effective" user id in a user or process context, and the owner permissions in a file context. Multiple UserId elements of this type may be used to specify a list of privileges.
4	current-group	The current group id (if applicable) being used by the user or process. On Unix systems, this would be the "real" group id, in general.
5	group-privs	Another group id the group or process has the ability to use, or a group id associated with a file permission. On Unix systems, this would be the "effective" group id in a group or process context, and the group permissions in a file context. On BSD-derived Unix systems, multiple UserId elements of this type would be used to include all the group ids on the "group list".

	6	other-privs	Not used in a user, group, or process context, only used in the file context. The file permissions assigned to users who do not match either the user or group permissions on the file. On Unix systems, this would be the "world" permissions.

tty

Optional. STRING. The tty the user is using.

4.2.7.4. The Process Class

The Process class is used to describe processes being executed on sources, targets, and analyzers.

The Process class is composed of five aggregate classes, as shown in Figure 18.

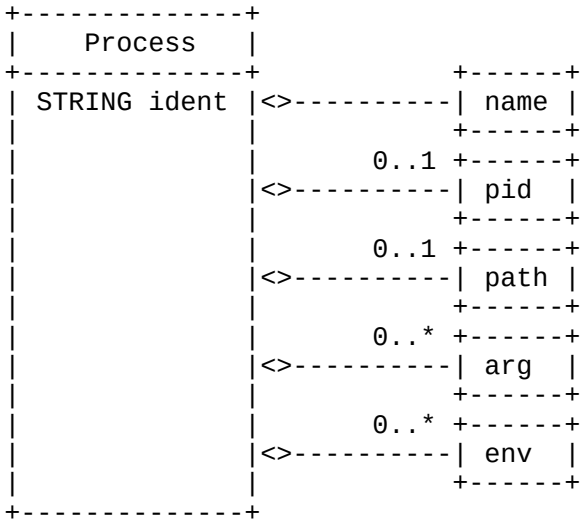

Figure 18: The Process Class

The aggregate classes that make up Process are:

name

Exactly one. STRING. The name of the program being executed. This is a short name; path and argument information are provided elsewhere.

pid

Zero or one. INTEGER. The process identifier of the process.

path

Zero or one. STRING. The full path of the program being executed.

arg

Zero or more. STRING. A command-line argument to the program. Multiple arguments may be specified (they are assumed to have occurred in the same order they are provided) with multiple uses of arg.

env

Zero or more. STRING. An environment string associated with the process; generally of the format "VARIABLE=value". Multiple environment strings may be specified with multiple uses of env.

This is represented in the IDMEF DTD as follows:

```
 <!ELEMENT Process                        (
    name, pid?, path?, arg*, env*
  )>
<!ATTLIST Process
    ident               CDATA                  '0'
    %attlist.global;
  >
```

The Process class has one attribute:

ident

Optional. A unique identifier for the process; see Section 3.2.9.

4.2.7.5. The Service Class

The Service class describes network services on sources and targets. It can identify services by name, port, and protocol. When Service occurs as an aggregate class of Source, it is understood that the service is one from which activity of interest is originating; and that the service is "attached" to the Node, Process, and User information also contained in Source. Likewise, when Service occurs as an aggregate class of Target, it is understood that the service is one to which activity of interest is being directed; and that the service is "attached" to the Node, Process, and User information also contained in Target. If Service occurs in both Source and Target, then information in both locations should be the same. If information is the same in both locations and implementers wish to carry it in only one location, they should specify it as an aggregate of the Target class.

The Service class is composed of four aggregate classes, as shown in Figure 19.

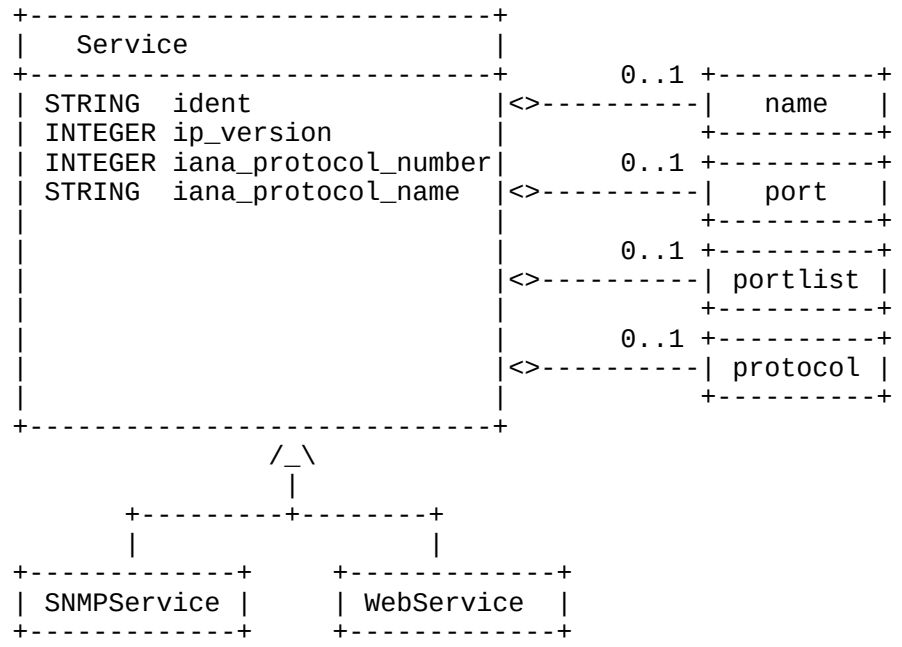

Figure 19: The Service Class

The aggregate classes that make up Service are:

name

Zero or one. STRING. The name of the service. Whenever possible, the name from the IANA list of well-known ports SHOULD be used.

port

Zero or one. INTEGER. The port number being used.

portlist

Zero or one. PORTLIST. A list of port numbers being used; see Section 3.2.8 for formatting rules. If a portlist is given, the iana_protocol_number and iana_protocol_name MUST apply to all the elements of the list.

protocol

Zero or one. STRING. Additional information about the protocol being used. The intent of the protocol field is to carry additional information related to the protocol being used when the <Service> attributes iana_protocol_number or/and iana_protocol_name are filed.

A Service MUST be specified as either (a) a name or a port or (b) a portlist. The protocol is optional in all cases, but no other combinations are permitted.

Service is represented in the IDMEF DTD as follows:

```
<!ELEMENT Service                        (
    (((name, port?) | (port, name?)) | portlist), protocol?,
    SNMPService?, WebService?
  )>
<!ATTLIST Service
    ident               CDATA                   '0'
ip_version           CDATA                   #IMPLIED
iana_protocol_number CDATA                   #IMPLIED
iana_protocol_name   CDATA                   #IMPLIED
    %attlist.global;
  >
```

The Service class has four attributes:

ident

Optional. A unique identifier for the service; see Section 3.2.9.

ip_version

Optional. INTEGER. The IP version number.

iana_protocol_number

Optional. INTEGER. The IANA protocol number.

iana_protocol_name

Optional. STRING. The IANA protocol name.

4.2.7.5.1. The WebService Class

The WebService class carries additional information related to web traffic.

The WebService class is composed of four aggregate classes, as shown in Figure 20.

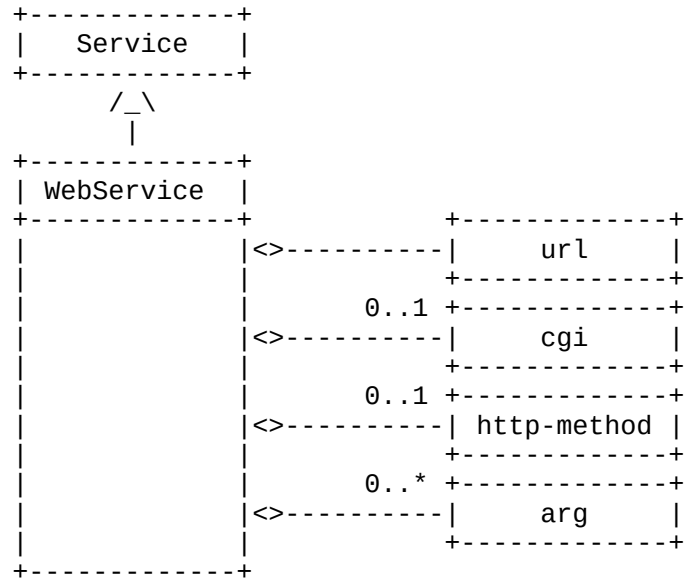

Figure 20: The WebService Class

The aggregate classes that make up WebService are:

url

Exactly one. STRING. The URL in the request.

cgi

Zero or one. STRING. The CGI script in the request, without arguments.

http-method

Zero or one. STRING. The HTTP method (PUT, GET) used in the request.

arg

Zero or more. STRING. The arguments to the CGI script.

This is represented in the IDMEF DTD as follows:

```
<!ELEMENT WebService                    (
    url, cgi?, http-method?, arg*
  )>
<!ATTLIST WebService
    %attlist.global;
  >
```

4.2.7.5.2. The SNMPService Class

The SNMPService class carries additional information related to SNMP traffic. The aggregate classes composing SNMPService must be interpreted as described in RFC 3411 [15] and RFC 3584 [16].

The SNMPService class is composed of eight aggregate classes, as shown in Figure 21.

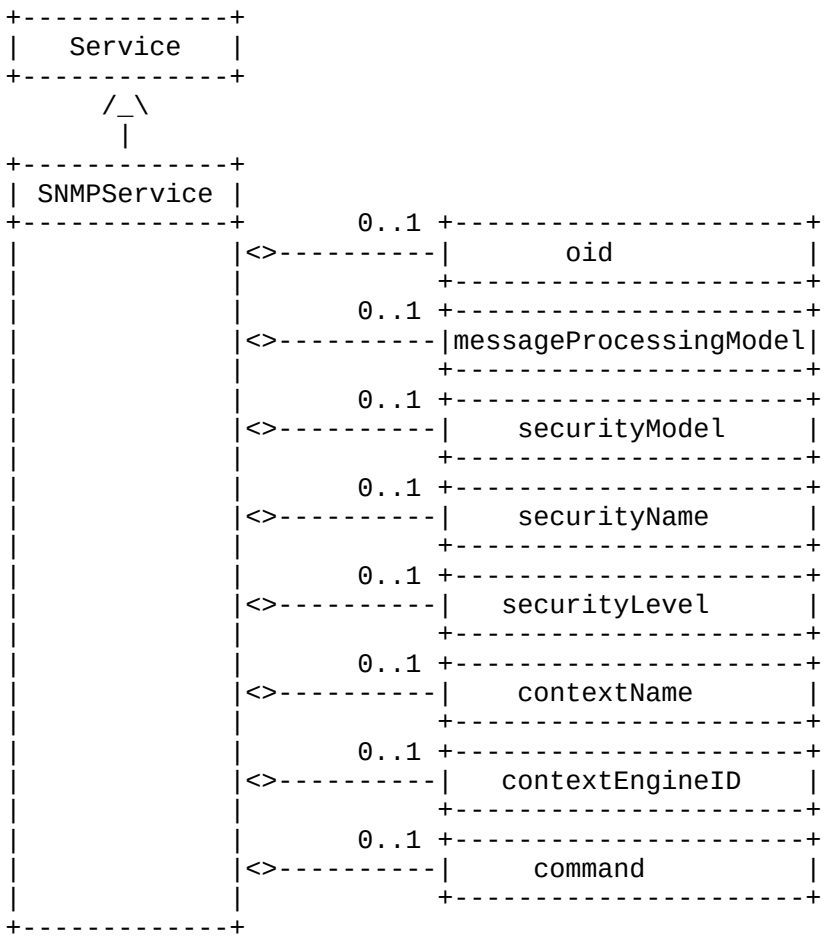

Figure 21: The SNMPService Class

The aggregate classes that make up SNMPService are:

oid

Zero or one. STRING. The object identifier in the request.

messageProcessingModel

Zero or one. INTEGER. The SNMP version, typically 0 for SNMPv1, 1 for SNMPv2c, 2 for SNMPv2u and SNMPv2*, and 3 for SNMPv3; see RFC 3411 [15] Section 5 for appropriate values.

securityModel

Zero or one. INTEGER. The identification of the security model in use, typically 0 for any, 1 for SNMPv1, 2 for SNMPv2c, and 3 for USM; see RFC 3411 [15] Section 5 for appropriate values.

securityName

Zero or one. STRING. The object's security name; see RFC 3411 [15] Section 3.2.2.

securityLevel

Zero or one. INTEGER. The security level of the SNMP request; see RFC 3411 [15] Section 3.4.3.

contextName

Zero or one. STRING. The object's context name; see RFC 3411 [15] Section 3.3.3.

contextEngineID

Zero or one. STRING. The object's context engine identifier; see RFC 3411 [15] Section 3.3.2.

command

Zero or one. STRING. The command sent to the SNMP server (GET, SET, etc.).

If other fields of an SNMP message are available and should be incorporated in the IDMEF alert, they must be located in the additionaldata structure with the meaning being an object definition defined in RFC 3411 [15] Section 5 and the value located within the additionaldata payload.

This is represented in the IDMEF DTD as follows:

```
<!ELEMENT SNMPService                    (
    oid?, messageProcessingModel?, securityModel?, securityName?,
    securityLevel?, contextName?, contextEngineID?, command?
  )>
<!ATTLIST SNMPService
    %attlist.global;
  >
```

4.2.7.6. The File Class

The File class provides specific information about a file or other file-like object that has been created, deleted, or modified on the target. The description can provide either the file settings prior to the event or the file settings at the time of the event, as specified using the "category" attribute.

The File class is composed of eleven aggregate classes, as shown in Figure 22.

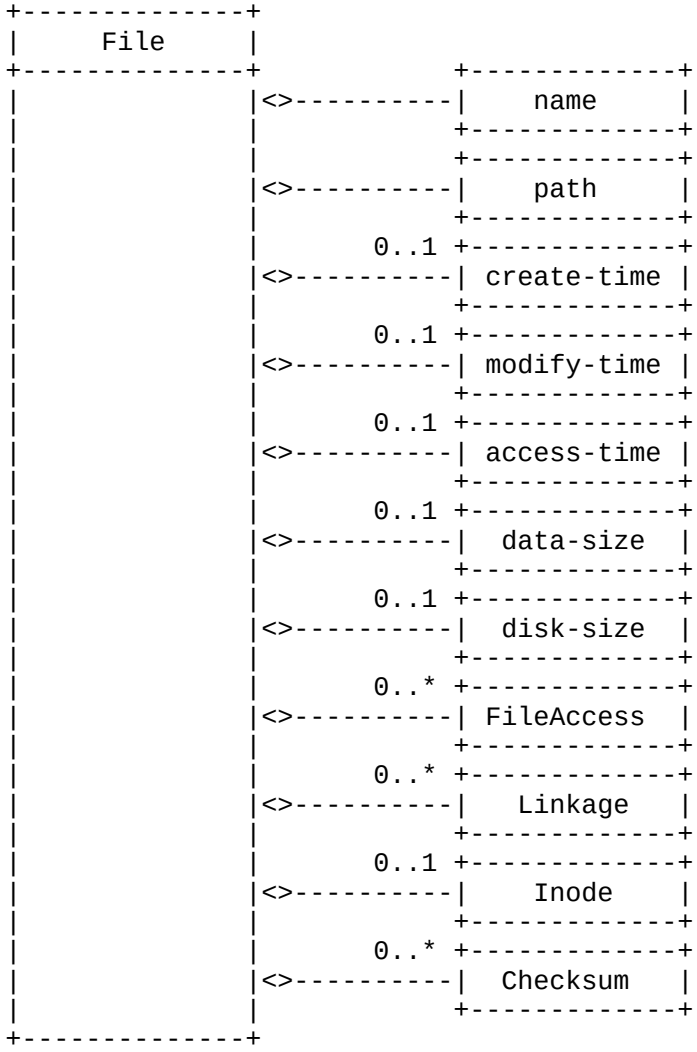

Figure 22: The File Class

The aggregate classes that make up File are:

name

Exactly one. STRING. The name of the file to which the alert applies, not including the path to the file.

path

Exactly one. STRING. The full path to the file, including the name. The path name should be represented in as "universal" a manner as possible, to facilitate processing of the alert.

For Windows systems, the path should be specified using the Universal Naming Convention (UNC) for remote files, and using a drive letter for local files (e.g., "C:\boot.ini"). For Unix systems, paths on network file systems should use the name of the mounted resource instead of the local mount point (e.g., "fileserver:/usr/local/bin/foo"). The mount point can be provided using the <Linkage> element.

create-time

Zero or one. DATETIME. Time the file was created. Note that this is *not* the Unix "st_ctime" file attribute (which is not file creation time). The Unix "st_ctime" attribute is contained in the "Inode" class.

modify-time

Zero or one. DATETIME. Time the file was last modified.

access-time

Zero or one. DATETIME. Time the file was last accessed.

data-size

Zero or one. INTEGER. The size of the data, in bytes. Typically what is meant when referring to file size. On Unix UFS file systems, this value corresponds to stat.st_size. On Windows NTFS, this value corresponds to Valid Data Length (VDL).

disk-size

Zero or one. INTEGER. The physical space on disk consumed by the file, in bytes. On Unix UFS file systems, this value corresponds to 512 * stat.st_blocks. On Windows NTFS, this value corresponds to End of File (EOF).

FileAccess

Zero or more. Access permissions on the file.

Linkage

Zero or more. File system objects to which this file is linked (other references for the file).

Inode

Zero or one. Inode information for this file (relevant to Unix).

Checksum

Zero or more. Checksum information for this file.

This is represented in the IDMEF DTD as follows:

```
<!ENTITY % attvals.filecat              "
    ( current | original )
  ">

<!ELEMENT File                          (
    name, path, create-time?, modify-time?, access-time?,
    data-size?, disk-size?, FileAccess*, Linkage*, Inode?,
    Checksum*
  )>
<!ATTLIST File
    ident               CDATA                   '0'
    category            %attvals.filecat;       #REQUIRED
    fstype              CDATA                   #IMPLIED
    file-type           CDATA                   #IMPLIED
    %attlist.global;
  >
```

The File class has four attributes (one required and three optional):

ident

Optional. A unique identifier for this file; see Section 3.2.9.

category

Required. The context for the information being provided. The permitted values are shown below. There is no default value. (See also Section 10.)

Rank	Keyword	Description
0	current	The file information is from after the reported change
1	original	The file information is from before the reported change

fstype

Optional. The type of file system the file resides on. This attribute governs how path names and other attributes are interpreted.

Rank	Keyword	Description
0	ufs	Berkeley Unix Fast File System
1	efs	Linux "efs" file system
2	nfs	Network File System
3	afs	Andrew File System
4	ntfs	Windows NT File System
5	fat16	16-bit Windows FAT File System
6	fat32	32-bit Windows FAT File System
7	pcfs	"PC" (MS-DOS) file system on CD-ROM
8	joliet	Joliet CD-ROM file system
9	iso9660	ISO 9660 CD-ROM file system

file-type

Optional. The type of file, as a mime-type.

4.2.7.6.1. The FileAccess Class

The FileAccess class represents the access permissions on a file. The representation is intended to be useful across operating systems.

The FileAccess class is composed of two aggregate classes, as shown in Figure 23.

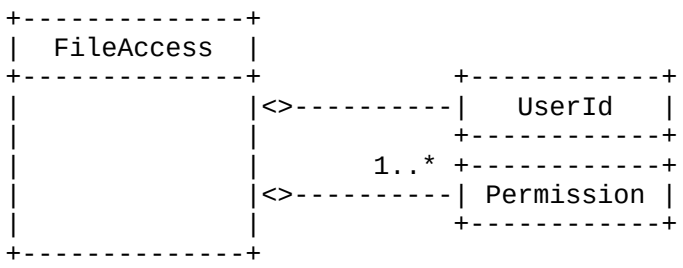

Figure 23: The FileAccess Class

The aggregate classes that make up FileAccess are:

UserId

Exactly one. The user (or group) to which these permissions apply. The value of the "type" attribute must be "user-privs", "group-privs", or "other-privs" as appropriate. Other values for "type" MUST NOT be used in this context.

Permission

One or more. ENUM. Level of access allowed. The permitted values are shown below. There is no default value. (See also Section 10.)

Rank	Keyword	Description
0	noAccess	No access at all is allowed for this user
1	read	This user has read access to the file
2	write	This user has write access to the file
3	execute	This user has the ability to execute the file
4	search	This user has the ability to search this file (applies to "execute" permission on directories in Unix)
5	delete	This user has the ability to delete this file
6	executeAs	This user has the ability to execute this file as another user
7	changePermissions	This user has the ability to change the access permissions on this file
8	takeOwnership	This user has the ability to take ownership of this file

The "changePermissions" and "takeOwnership" strings represent those concepts in Windows. On Unix, the owner of the file always has "changePermissions" access, even if no other access is allowed for that user. "Full Control" in Windows is represented by enumerating the permissions it contains. The "executeAs" string represents the set-user-id and set-group-id features in Unix.

This is represented in the IDMEF DTD as follows:

```
<!ELEMENT Permission EMPTY >
<!ATTLIST Permission
    perms              %attvals.fileperm;      #REQUIRED
    %attlist.global;
>

<!ENTITY % attvals.fileperm "( noAccess | read | write | execute |
  search | delete | executeAs | changePermissions |
  takeOwnership)" >
```

4.2.7.6.2. The Linkage Class

The Linkage class represents file system connections between the file described in the <File> element and other objects in the file system. For example, if the <File> element is a symbolic link or shortcut, then the <Linkage> element should contain the name of the object the link points to. Further information can be provided about the object in the <Linkage> element with another <File> element, if appropriate.

The Linkage class is composed of three aggregate classes, as shown in Figure 24.

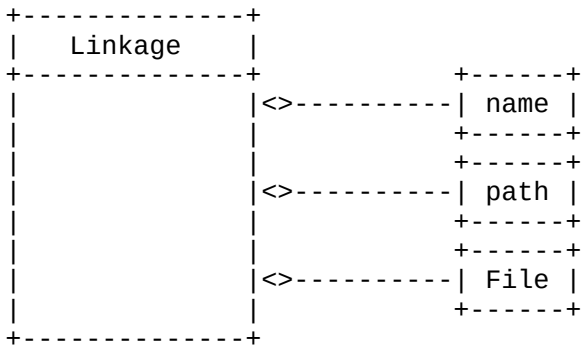

Figure 24: The Linkage Class

The aggregate classes that make up Linkage are:

name

Exactly one. STRING. The name of the file system object, not including the path.

path

Exactly one. STRING. The full path to the file system object, including the name. The path name should be represented in as "universal" a manner as possible, to facilitate processing of the alert.

File

Exactly one. A <File> element may be used in place of the <name> and <path> elements if additional information about the file is to be included.

This is represented in the IDMEF DTD as follows:

```
<!ENTITY % attvals.linkcat              "
    ( hard-link | mount-point | reparse-point | shortcut | stream |
      symbolic-link )
  ">

<!ELEMENT Linkage                       (
    (name, path) | File
  )>
<!ATTLIST Linkage
    category            %attvals.linkcat;       #REQUIRED
    %attlist.global;
  >
```

The Linkage class has one attribute:

category

The type of object that the link describes. The permitted values are shown below. There is no default value. (See also Section 10.)

Rank	Keyword	Description
0	hard-link	The <name> element represents another name for this file. This information may be more easily obtainable on NTFS file systems than others.
1	mount-point	An alias for the directory specified by the parent's <name> and <path> elements.
2	reparse-point	Applies only to Windows; excludes symbolic links and mount points, which are specific types of reparse points.
3	shortcut	The file represented by a Windows "shortcut". A shortcut is distinguished from a symbolic link because of the difference in their contents, which may be of importance to the manager.
4	stream	An Alternate Data Stream (ADS) in Windows; a fork on MacOS. Separate file system entity that is considered an extension of the main <File>.
5	symbolic-link	The <name> element represents the file to which the link points.

4.2.7.6.3. The Inode Class

The Inode class is used to represent the additional information contained in a Unix file system i-node.

The Inode class is composed of six aggregate classes, as shown in Figure 25.

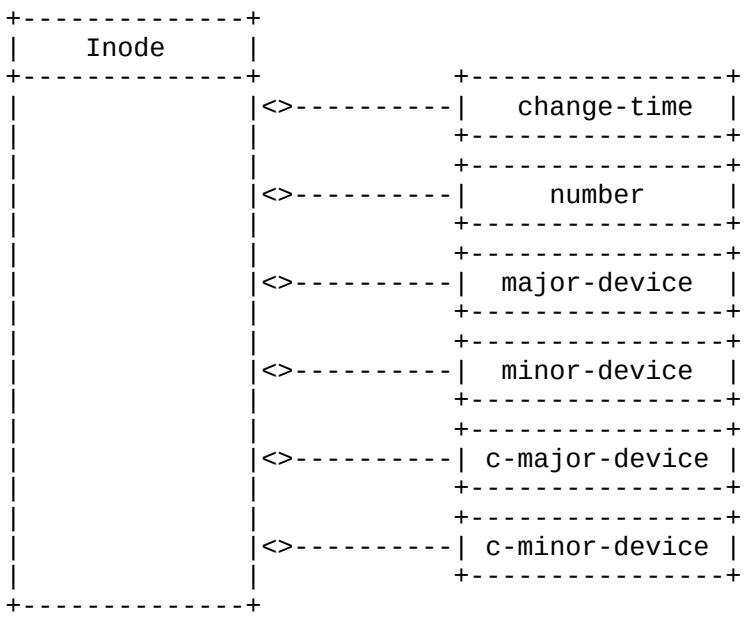

Figure 25: The Inode Class

The aggregate classes that make up Inode are:

change-time

Zero or one. DATETIME. The time of the last inode change, given by the st_ctime element of "struct stat".

number

Zero or one. INTEGER. The inode number.

major-device

Zero or one. INTEGER. The major device number of the device the file resides on.

minor-device

Zero or one. INTEGER. The minor device number of the device the file resides on.

c-major-device

Zero or one. INTEGER. The major device of the file itself, if it is a character special device.

c-minor-device

Zero or one. INTEGER. The minor device of the file itself, if it is a character special device.

Note that <number>, <major-device>, and <minor-device> must be given together, and the <c-major-device> and <c-minor-device> must be given together.

This is represented in the IDMEF DTD as follows:

```
<!ELEMENT Inode                          (
    change-time?, (number, major-device, minor-device)?,
    (c-major-device, c-minor-device)?
  )>
<!ATTLIST Inode
    %attlist.global;
  >
```

4.2.7.6.4. The Checksum Class

The Checksum class represents checksum information associated with the file. This checksum information can be provided by file integrity checkers, among others.

The checksum class is composed of two aggregate classes, as shown in Figure 26.

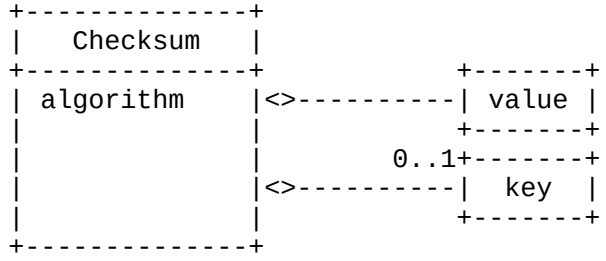

Figure 26: The Checksum Class

The aggregate classes that make up Checksum are:

value

Exactly one. STRING. The value of the checksum.

key

Zero or one. STRING. The key to the checksum, if appropriate.

This is represented in the IDMEF DTD as follows:

```
<!ENTITY % attvals.checksumalgos        "
      ( MD4 | MD5 | SHA1 | SHA2-256 | SHA2-384 | SHA2-512 | CRC-32 |
        Haval | Tiger | Gost )
  ">

<!ELEMENT Checksum                      (
    value, key?
  )>
<!ATTLIST Checksum
    algorithm          %attvals.checksumalgos; #REQUIRED
    %attlist.global;
  >
```

The Checksum class has one attribute:

algorithm

The cryptographic algorithm used for the computation of the checksum. The permitted values are shown below. There is no default value. (See also Section 10.)

Rank	Keyword	Description
0	MD4	The MD4 algorithm.
1	MD5	The MD5 algorithm.
2	SHA1	The SHA1 algorithm.
3	SHA2-256	The SHA2 algorithm with 256 bits length.
4	SHA2-384	The SHA2 algorithm with 384 bits length.
5	SHA2-512	The SHA2 algorithm with 512 bits length.
6	CRC-32	The CRC algorithm with 32 bits length.
7	Haval	The Haval algorithm.
8	Tiger	The Tiger algorithm.
9	Gost	The Gost algorithm.

5. Extending the IDMEF

As intrusion detection systems evolve, the IDMEF data model and DTD will have to evolve along with them. To allow new features to be added as they are developed, both the data model and the DTD can be extended as described in this section. As these extensions mature, they can then be incorporated into future versions of the specification.

5.1. Extending the Data Model

There are two mechanisms for extending the IDMEF data model, inheritance and aggregation:

- Inheritance denotes a superclass/subclass type of relationship where the subclass inherits all the attributes, operations, and

relationships of the superclass. This type of relationship is also called a "is-a" or "kind-of" relationship. Subclasses may have additional attributes or operations that apply only to the subclass and not to the superclass.

- Aggregation is a form of association in which the whole is related to its parts. This type of relationship is also referred to as a "part-of" relationship. In this case, the aggregate class contains all of its own attributes and as many of the attributes associated with its parts as required and specified by occurrence indicators.

Of the two mechanisms, inheritance is preferred, because it preserves the existing data model structure and also preserves the operations (methods) executed on the classes of the structure.

Note that the rules for extending the IDMEF DTD (see below) set limits on the places where extensions to the data model may be made.

5.2. Extending the IDMEF DTD

There are two ways to extend the IDMEF DTD:

1. The AdditionalData class (see Section 4.2.4.6) allows implementors to include arbitrary "atomic" data items (integers, strings, etc.) in an Alert or Heartbeat message. This approach SHOULD be used whenever possible. See Section 7.4 and Section 7.5.

2. The AdditionalData class allows implementors to extend the IDMEF DTD with additional DTD "modules" that describe arbitrarily complex data types and relationships. The remainder of this section describes this extension method.

To extend the IDMEF DTD with a new DTD "module", the following steps MUST be followed:

1. The document declaration MUST define a DTD location that defines the namespace and contains the location of the extension DTD, and then reference that namespace.

2. Multiple extensions may be included by defining multiple namespaces and DTD locations, and referencing them.

3. Extension DTDs MUST declare all of their elements and attributes in a separate XML namespace. Extension DTDs MUST NOT declare any elements or attributes in the "idmef" or default namespaces.

4. Extensions MUST only be included in IDMEF Alert and Heartbeat messages under an <AdditionalData> element whose "type" attribute contains the value "xml". For example:

In this example, the "vendorco" namespace is defined and then referenced, causing the DTD for the extension to be read by the XML parser.

```
<idmef:IDMEF-Message version="1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:idmef="http://iana.org/idmef"
  xmlns:vendorco="http://vendor.com/idmef"
xsi:schemaLocation="http://vendor.com/idmef http://v.com/vidmef.xsd">

 <idmef:Alert messageid="...">
      ...
 <idmef:AdditionalData type="xml" meaning="VendorExtension">
  <idmef:xml>
   <vendorco:TestVendor a="attribute of example"
    xmlns:vendorco="http://vendor.com/idmef"
xsi:schemaLocation="http://vendor.com/idmef http://v.com/vidmef.xsd">
    <vendorco:content>content element of example</vendorco:content>
   </vendorco:TestVendor>
  </idmef:xml>
 </idmef:AdditionalData>
 </idmef:Alert>
</idmef:IDMEF-Message>
```

See Section 7.8 for another example of extending the IDMEF DTD.

6. Special Considerations

This section discusses some of the special considerations that must be taken into account by implementors of the IDMEF.

6.1. XML Validity and Well-Formedness

It is expected that IDMEF-compliant applications will not normally include the IDMEF DTD itself in their communications. Instead, the DTD will be referenced in the document type definition in the IDMEF message. Such IDMEF documents will be well-formed and valid as defined in [3].

Other IDMEF documents will be specified that do not include the document prolog (e.g., entries in an IDMEF-format database). Such IDMEF documents will be well-formed but not valid.

Generally, well-formedness implies that a document has a single element that contains everything else (e.g., "<Book>") and that all the other elements nest nicely within each other without any overlapping (e.g., a "chapter" does not start in the middle of another "chapter").

Validity further implies that not only is the document well-formed, but it also follows specific rules (contained in the Document Type Definition) about which elements are "legal" in the document, how those elements nest within other elements, and so on (e.g., a "chapter" does not begin in the middle of a "title"). A document cannot be valid unless it references a DTD.

XML processors are required to be able to parse any well-formed document, valid or not. The purpose of validation is to make the processing of that document (what's done with the data after it's parsed) easier. Without validation, a document may contain elements in nonsense order, elements "invented" by the author that the processing application doesn't understand, and so forth.

IDMEF documents MUST be well-formed. IDMEF documents SHOULD be valid whenever both possible and practical.

6.2. Unrecognized XML Tags

On occasion, an IDMEF-compliant application may receive a well-formed, or even well-formed and valid, IDMEF message containing tags that it does not understand. The tags may be either:

- Recognized as "legitimate" (a valid document), but the application does not know the semantic meaning of the element's content; or
- Not recognized at all.

IDMEF-compliant applications MUST continue to process IDMEF messages that contain unknown tags, provided that such messages meet the well-formedness requirement of Section 6.1. It is up to the individual application to decide how to process (or ignore) any content from the unknown elements(s).

6.3. Analyzer-Manager Time Synchronization

Synchronization of time-of-day clocks between analyzers and managers is outside the scope of this document. However, the following comments and suggestions are offered:

1. Whenever possible, all analyzers and managers should have their time-of-day clocks synchronized to an external source such as NTP [7] or SNTP [8] Global Positioning System (GPS), Geosynchronous Operational Environmental Satellite (GOES), NIST radio station WWV clocks, or some other reliable time standard.

2. When external time synchronization is not possible, the IDMEF provides the <AnalyzerTime> element, which may be used to perform rudimentary time synchronization (see below).

3. IDMEF-compliant applications SHOULD permit the user to enable/disable the <AnalyzerTime> method of time synchronization as a configuration option.

A number of caveats apply to the use of <AnalyzerTime> for time synchronization:

1. <AnalyzerTime> works best in a "flat" environment where analyzers report up to a single level of managers. When a tree topology of high-level managers, intermediate relays, and analyzers is used, the problem becomes more complex.

2. When intermediate message relays (managers or otherwise) are involved, two scenarios are possible:

   * The intermediaries may forward entire IDMEF messages, or may perform aggregation or correlation, but MUST NOT inject delay. In this case, time synchronization is end-to-end between the analyzer and the highest-level manager.

   * The intermediaries may inject delay, due to storage or additional processing. In this case, time synchronization MUST be performed at each hop. This means each intermediary must decompose the IDMEF message, adjust all time values, and then reconstruct the message before sending it on.

3. When the environment is mixed, with some analyzers and managers using external time synchronization and some not, all managers and intermediaries must perform <AnalyzerTime> synchronization. This is because determining whether or not compensation is actually needed between two parties rapidly becomes very complex, and requires knowledge of other parts of the topology.

4. If an alert can take alternate paths, or be stored in multiple locations, the recorded times may be different depending on the path taken.

The above being said, <AnalyzerTime> synchronization is probably still better than nothing in many environments. To implement this type of synchronization, the following procedure is suggested:

1. When an analyzer or manager sends an IDMEF message, it should place the current value of its time-of-day clock in an <AnalyzerTime> element. This should occur as late as possible in the message transmission process, ideally right before the message is "put on the wire".

2. When a manager receives an IDMEF message, it should compute the difference between its own time-of-day clock and the time in the <AnalyzerTime> element of the message. This difference should then be used to adjust the times in the <CreateTime> and <DetectTime> elements (NTP timestamps should also be adjusted).

3. If the manager is an intermediary and sends the IDMEF message on to a higher-level manager, and hop-by-hop synchronization is in effect, it should regenerate the <AnalyzerTime> value to contain the value of its own time-of-day clock.

6.4. NTP Timestamp Wrap-Around

From [8]:

> Note that, since some time in 1968 (second 2,147,483,648) the most significant bit (bit 0 of the integer part) has been set and that the 64-bit field will overflow some time in 2036 (second 4,294,967,296). Should NTP or SNTP be in use in 2036, some external means will be necessary to qualify time relative to 1900 and time relative to 2036 (and other multiples of 136 years). There will exist a 200-picosecond interval, henceforth ignored, every 136 years when the 64-bit field will be 0, which by convention is interpreted as an invalid or unavailable timestamp.

IDMEF-compliant applications MUST NOT send a zero-valued NTP timestamp unless they mean to indicate that it is invalid or unavailable. If an IDMEF-compliant application must send an IDMEF message at the time of rollover, the application should wait for 200 picoseconds until the timestamp will have a non-zero value.

Also from [8]:

> As the NTP timestamp format has been in use for the last 17 years, it remains a possibility that it will be in use 40 years from now when the seconds field overflows. As it is probably inappropriate to archive NTP timestamps before bit 0 was set in 1968, a

convenient way to extend the useful life of NTP timestamps is the following convention:

If bit 0 is set, the UTC time is in the range 1968-2036 and UTC time is reckoned from 0h 0m 0s UTC on 1 January 1900.

If bit 0 is not set, the time is in the range 2036-2104 and UTC time is reckoned from 6h 28m 16s UTC on 7 February 2036.

Note that when calculating the correspondence, 2000 is not a leap year. Note also that leap seconds are not counted in the reckoning.

IDMEF-compliant applications in use after 2036-02-07T06:28:16Z MUST adhere to the above convention.

6.5. Digital Signatures

Standard XML digital signature processing rules and syntax are specified in [13]. XML Signatures provide integrity, message authentication, and/or signer authentication services for data of any type, whether located within the XML that includes the signature or elsewhere.

The IDMEF requirements document [2] assigns responsibility for message integrity and authentication to the communications protocol, not the message format. However, in situations where IDMEF messages are exchanged over other, less secure protocols, or in cases where the digital signatures must be archived for later use, the inclusion of digital signatures within an IDMEF message itself may be desirable.

Specifications for the use of digital signatures within IDMEF messages are outside the scope of this document. However, if such functionality is needed, use of the XML Signature standard is RECOMMENDED.

7. Examples

The examples shown in this section demonstrate how the IDMEF is used to encode alert data. These examples are for illustrative purposes only, and do not necessarily represent the only (or even the "best") way to encode these particular alerts. These examples should not be taken as guidelines on how alerts should be classified.

7.1. Denial-of-Service Attacks

The following examples show how some common denial-of-service attacks could be represented in the IDMEF.

7.1.1. The "teardrop" Attack

Network-based detection of the "teardrop" attack. This shows the basic format of an alert.

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message xmlns:idmef="http://iana.org/idmef"
                     version="1.0">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="hq-dmz-analyzer01">
      <idmef:Node category="dns">
        <idmef:location>Headquarters DMZ Network</idmef:location>
        <idmef:name>analyzer01.example.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc723b45.0xef449129">
      2000-03-09T10:01:25.93464-05:00
    </idmef:CreateTime>
    <idmef:Source ident="a1b2c3d4">
      <idmef:Node ident="a1b2c3d4-001" category="dns">
        <idmef:name>badguy.example.net</idmef:name>
        <idmef:Address ident="a1b2c3d4-002"
                       category="ipv4-net-mask">
          <idmef:address>192.0.2.50</idmef:address>
          <idmef:netmask>255.255.255.255</idmef:netmask>
        </idmef:Address>
      </idmef:Node>
    </idmef:Source>
    <idmef:Target ident="d1c2b3a4">
      <idmef:Node ident="d1c2b3a4-001" category="dns">
        <idmef:Address category="ipv4-addr-hex">
          <idmef:address>0xde796f70</idmef:address>
        </idmef:Address>
      </idmef:Node>
    </idmef:Target>
    <idmef:Classification text="Teardrop detected">
      <idmef:Reference origin="bugtraqid">
        <idmef:name>124</idmef:name>
        <idmef:url>http://www.securityfocus.com/bid/124</idmef:url>
      </idmef:Reference>
    </idmef:Classification>
  </idmef:Alert>
```

```
   </idmef:IDMEF-Message>
```

7.1.2. The "ping of death" Attack

Network-based detection of the "ping of death" attack. Note the identification of multiple targets, and the identification of the source as a spoofed address.

NOTE: The URL has been cut to fit the IETF formating requirements.

```
   <?xml version="1.0" encoding="UTF-8"?>

   <idmef:IDMEF-Message version="1.0"
                        xmlns:idmef="http://iana.org/idmef">
     <idmef:Alert messageid="abc123456789">
       <idmef:Analyzer analyzerid="bc-sensor01">
         <idmef:Node category="dns">
           <idmef:name>sensor.example.com</idmef:name>
         </idmef:Node>
       </idmef:Analyzer>
       <idmef:CreateTime ntpstamp="0xbc71f4f5.0xef449129">
         2000-03-09T10:01:25.93464Z
       </idmef:CreateTime>
       <idmef:Source ident="a1a2" spoofed="yes">
         <idmef:Node ident="a1a2-1">
           <idmef:Address ident="a1a2-2" category="ipv4-addr">
             <idmef:address>192.0.2.200</idmef:address>
           </idmef:Address>
         </idmef:Node>
       </idmef:Source>
       <idmef:Target ident="b3b4">
         <idmef:Node>
           <idmef:Address ident="b3b4-1" category="ipv4-addr">
             <idmef:address>192.0.2.50</idmef:address>
           </idmef:Address>
         </idmef:Node>
       </idmef:Target>
       <idmef:Target ident="c5c6">
         <idmef:Node ident="c5c6-1" category="nisplus">
           <idmef:name>lollipop</idmef:name>
         </idmef:Node>
       </idmef:Target>
       <idmef:Target ident="d7d8">
         <idmef:Node ident="d7d8-1">
           <idmef:location>Cabinet B10</idmef:location>
           <idmef:name>Cisco.router.b10</idmef:name>
         </idmef:Node>
       </idmef:Target>
```

```
      <idmef:Classification text="Ping-of-death detected">
        <idmef:Reference origin="cve">
          <idmef:name>CVE-1999-128</idmef:name>
          <idmef:url>http://www.cve.mitre.org/cgi-bin/
          cvename.cgi?name=CVE-1999-128</idmef:url>
        </idmef:Reference>
      </idmef:Classification>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

7.2. Port Scanning Attacks

The following examples show how some common port scanning attacks could be represented in the IDMEF.

7.2.1. Connection to a Disallowed Service

Host-based detection of a policy violation (attempt to obtain information via "finger"). Note the identification of the target service, as well as the originating user (obtained, e.g., through RFC 1413 [11]).

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="bc-sensor01">
      <idmef:Node category="dns">
        <idmef:name>sensor.example.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc72541d.0x00000000">
      2000-03-09T18:47:25+02:00
    </idmef:CreateTime>
    <idmef:Source ident="a123">
      <idmef:Node ident="a123-01">
        <idmef:Address ident="a123-02" category="ipv4-addr">
          <idmef:address>192.0.2.200</idmef:address>
        </idmef:Address>
      </idmef:Node>
      <idmef:User ident="q987-03" category="os-device">
        <idmef:UserId ident="q987-04" type="target-user">
          <idmef:name>badguy</idmef:name>
        </idmef:UserId>
      </idmef:User>
      <idmef:Service ident="a123-03">
        <idmef:port>31532</idmef:port>
```

```
        </idmef:Service>
      </idmef:Source>
      <idmef:Target ident="z456">
        <idmef:Node ident="z456-01" category="nis">
          <idmef:name>myhost</idmef:name>
          <idmef:Address ident="z456-02" category="ipv4-addr">
            <idmef:address>192.0.2.50</idmef:address>
          </idmef:Address>
        </idmef:Node>
        <idmef:Service ident="z456-03">
          <idmef:name>finger</idmef:name>
          <idmef:port>79</idmef:port>
        </idmef:Service>
      </idmef:Target>
      <idmef:Classification text="Portscan">
        <idmef:Reference origin="vendor-specific">
          <idmef:name>finger</idmef:name>
          <idmef:url>http://www.vendor.com/finger</idmef:url>
        </idmef:Reference>
        <idmef:Reference origin="vendor-specific"
                         meaning="general documentation">
          <idmef:name>Distributed attack</idmef:name>
          <idmef:url>http://www.vendor.com/distributed</idmef:url>
        </idmef:Reference>
      </idmef:Classification>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

7.2.2. Simple Port Scanning

Network-based detection of a port scan. This shows detection by a single analyzer; see Section 7.5 for the same attack as detected by a correlation engine. Note the use of <portlist> to show the ports that were scanned.

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="hq-dmz-analyzer62">
      <idmef:Node category="dns">
        <idmef:location>Headquarters Web Server</idmef:location>
        <idmef:name>analyzer62.example.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc72b2b4.0x00000000">
      2000-03-09T15:31:00-08:00
```

```
      </idmef:CreateTime>
      <idmef:Source ident="abc01">
        <idmef:Node ident="abc01-01">
          <idmef:Address ident="abc01-02" category="ipv4-addr">
            <idmef:address>192.0.2.200</idmef:address>
          </idmef:Address>
        </idmef:Node>
      </idmef:Source>
      <idmef:Target ident="def01">
        <idmef:Node ident="def01-01" category="dns">
          <idmef:name>www.example.com</idmef:name>
          <idmef:Address ident="def01-02" category="ipv4-addr">
            <idmef:address>192.0.2.50</idmef:address>
          </idmef:Address>
        </idmef:Node>
        <idmef:Service ident="def01-03">
          <idmef:portlist>5-25,37,42,43,53,69-119,123-514
          </idmef:portlist>
        </idmef:Service>
      </idmef:Target>
      <idmef:Classification text="simple portscan">
        <idmef:Reference origin="vendor-specific">
          <idmef:name>portscan</idmef:name>
          <idmef:url>http://www.vendor.com/portscan</idmef:url>
        </idmef:Reference>
      </idmef:Classification>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

7.3. Local Attacks

The following examples show how some common local host attacks could be represented in the IDMEF.

7.3.1. The "loadmodule" Attack

Host-based detection of the "loadmodule" exploit. This attack involves tricking the "loadmodule" program into running another program; since "loadmodule" is set-user-id "root", the executed program runs with super-user privileges. Note the use of <User> and <Process> to identify the user attempting the exploit and how he's doing it.

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
```

```
      <idmef:Analyzer analyzerid="bc-fs-sensor13">
        <idmef:Node category="dns">
          <idmef:name>fileserver.example.com</idmef:name>
        </idmef:Node>
        <idmef:Process>
          <idmef:name>monitor</idmef:name>
          <idmef:pid>8956</idmef:pid>
          <idmef:arg>monitor</idmef:arg>
          <idmef:arg>-d</idmef:arg>
          <idmef:arg>-m</idmef:arg>
          <idmef:arg>idmanager.example.com</idmef:arg>
          <idmef:arg>-l</idmef:arg>
          <idmef:arg>/var/logs/idlog</idmef:arg>
        </idmef:Process>
      </idmef:Analyzer>
      <idmef:CreateTime ntpstamp="0xbc7221c0.0x4ccccccc">
        2000-03-09T08:12:32.3-05:00
      </idmef:CreateTime>
      <idmef:Source ident="a1a2">
        <idmef:User ident="a1a2-01" category="os-device">
          <idmef:UserId ident="a1a2-02"
                        type="original-user">
            <idmef:name>joe</idmef:name>
            <idmef:number>13243</idmef:number>
          </idmef:UserId>
        </idmef:User>
        <idmef:Process ident="a1a2-03">
          <idmef:name>loadmodule</idmef:name>
          <idmef:path>/usr/openwin/bin</idmef:path>
        </idmef:Process>
      </idmef:Source>
      <idmef:Target ident="z3z4">
        <idmef:Node ident="z3z4-01" category="dns">
          <idmef:name>fileserver.example.com</idmef:name>
        </idmef:Node>
      </idmef:Target>
      <idmef:Classification text="Loadmodule attack"
                            ident="loadmodule">
        <idmef:Reference origin="bugtraqid">
          <idmef:name>33</idmef:name>
          <idmef:url>http://www.securityfocus.com</idmef:url>
        </idmef:Reference>
      </idmef:Classification>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

The Intrusion Detection System (IDS) could also indicate that the target user is the "root" user, and show the attempted command; the alert might then look like:

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="bc-fs-sensor13">
      <idmef:Node category="dns">
        <idmef:name>fileserver.example.com</idmef:name>
      </idmef:Node>
      <idmef:Process>
        <idmef:name>monitor</idmef:name>
        <idmef:pid>8956</idmef:pid>
        <idmef:arg>monitor</idmef:arg>
        <idmef:arg>-d</idmef:arg>
        <idmef:arg>-m</idmef:arg>
        <idmef:arg>idmanager.example.com</idmef:arg>
        <idmef:arg>-l</idmef:arg>
        <idmef:arg>/var/logs/idlog</idmef:arg>
      </idmef:Process>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc7221c0.0x4ccccccc">
      2000-03-09T08:12:32.3-05:00
    </idmef:CreateTime>
    <idmef:Source ident="a1a2">
      <idmef:User ident="a1a2-01" category="os-device">
        <idmef:UserId ident="a1a2-02" type="original-user">
          <idmef:name>joe</idmef:name>
          <idmef:number>13243</idmef:number>
        </idmef:UserId>
      </idmef:User>
      <idmef:Process ident="a1a2-03">
        <idmef:name>loadmodule</idmef:name>
        <idmef:path>/usr/openwin/bin</idmef:path>
      </idmef:Process>
    </idmef:Source>
    <idmef:Target ident="z3z4">
      <idmef:Node ident="z3z4-01" category="dns">
        <idmef:name>fileserver.example.com</idmef:name>
      </idmef:Node>
      <idmef:User ident="z3z4-02" category="os-device">
        <idmef:UserId ident="z3z4-03" type="target-user">
          <idmef:name>root</idmef:name>
          <idmef:number>0</idmef:number>
        </idmef:UserId>
```

```
        </idmef:User>
        <idmef:Process ident="z3z4-04">
          <idmef:name>sh</idmef:name>
          <idmef:pid>25134</idmef:pid>
          <idmef:path>/bin/sh</idmef:path>
        </idmef:Process>
      </idmef:Target>
      <idmef:Classification text="Loadmodule attack"
                            ident="loadmodule">
      </idmef:Classification>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

Note that the identification of the classification is used.

7.3.2. The "phf" Attack

Network-based detection of the "phf" attack. Note the use of the <WebService> element to provide more details about this particular attack.

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="bc-sensor01">
      <idmef:Node category="dns">
        <idmef:name>sensor.example.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc71e980.0x00000000">
      2000-03-09T08:12:32-01:00
    </idmef:CreateTime>
    <idmef:Source ident="abc123">
      <idmef:Node ident="abc123-001">
        <idmef:Address ident="abc123-002"
                       category="ipv4-addr">
          <idmef:address>192.0.2.200</idmef:address>
        </idmef:Address>
      </idmef:Node>
      <idmef:Service ident="abc123-003">
        <idmef:port>21534</idmef:port>
      </idmef:Service>
    </idmef:Source>
    <idmef:Target ident="xyz789">
      <idmef:Node ident="xyz789-001" category="dns">
        <idmef:name>www.example.com</idmef:name>
```

```
          <idmef:Address ident="xyz789-002"
                         category="ipv4-addr">
            <idmef:address>192.0.2.100</idmef:address>
          </idmef:Address>
        </idmef:Node>
        <idmef:Service>
          <idmef:port>8080</idmef:port>
          <idmef:WebService>
            <idmef:url>
            http://www.example.com/cgi-bin/phf?/etc/group
            </idmef:url>
            <idmef:cgi>/cgi-bin/phf</idmef:cgi>
            <idmef:http-method>GET</idmef:http-method>
          </idmef:WebService>
        </idmef:Service>
      </idmef:Target>
      <idmef:Classification text="phf attack">
        <idmef:Reference origin="bugtraqid">
          <idmef:name>629</idmef:name>
          <idmef:url>
          http://www.securityfocus.com/bid/629
          </idmef:url>
        </idmef:Reference>
      </idmef:Classification>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

7.3.3. File Modification

Host-based detection of a race condition attack. Note the use of the <File> to provide information about the files that are used to perform the attack.

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert>
    <idmef:Analyzer analyzerid="bids-192.0.2.1"
                    ostype="Linux"
                    osversion="2.2.16-3">
      <idmef:Node category="hosts">
        <idmef:name>etude</idmef:name>
        <idmef:Address category="ipv4-addr">
          <idmef:address>192.0.2.1</idmef:address>
        </idmef:Address>
      </idmef:Node>
    </idmef:Analyzer>
```

```
      <idmef:CreateTime ntpstamp="0xbc71e980.0x00000000">
        2000-03-09T08:12:32-01:00
      </idmef:CreateTime>
      <idmef:Source spoofed="no">
        <idmef:Node>
          <idmef:location>console</idmef:location>
          <idmef:Address category="ipv4-addr">
            <idmef:address>192.0.2.1</idmef:address>
          </idmef:Address>
          </idmef:Node>
      </idmef:Source>
      <idmef:Target decoy="no">
        <idmef:Node>
          <idmef:location>local</idmef:location>
          <idmef:Address category="ipv4-addr">
            <idmef:address>192.0.2.1</idmef:address>
          </idmef:Address>
        </idmef:Node>
        <idmef:User category="os-device">
          <idmef:UserId type="original-user">
            <idmef:number>456</idmef:number>
          </idmef:UserId>
          <idmef:UserId type="current-user">
            <idmef:name>fred</idmef:name>
            <idmef:number>456</idmef:number>
          </idmef:UserId>
          <idmef:UserId type="user-privs">
            <idmef:number>456</idmef:number>
          </idmef:UserId>
        </idmef:User>
        <idmef:File category="current" fstype="tmpfs">
          <idmef:name>xxx000238483</idmef:name>
          <idmef:path>/tmp/xxx000238483</idmef:path>
          <idmef:FileAccess>
            <idmef:UserId type="user-privs">
              <idmef:name>alice</idmef:name>
              <idmef:number>777</idmef:number>
            </idmef:UserId>
            <idmef:permission perms="read" />
            <idmef:permission perms="write" />
            <idmef:permission perms="delete" />
            <idmef:permission perms="changePermissions" />
          </idmef:FileAccess>
          <idmef:FileAccess>
            <idmef:UserId type="group-privs">
              <idmef:name>user</idmef:name>
              <idmef:number>42</idmef:number>
            </idmef:UserId>
```

```
            <idmef:permission perms="read" />
            <idmef:permission perms="write" />
            <idmef:permission perms="delete" />
          </idmef:FileAccess>
          <idmef:FileAccess>
            <idmef:UserId type="other-privs">
              <idmef:name>world</idmef:name>
            </idmef:UserId>
            <idmef:permission perms="noAccess" />
          </idmef:FileAccess>
          <idmef:Linkage category="symbolic-link">
            <idmef:name>passwd</idmef:name>
            <idmef:path>/etc/passwd</idmef:path>
          </idmef:Linkage>
        </idmef:File>
      </idmef:Target>
      <idmef:Classification text="DOM race condition">
        <idmef:Reference origin="vendor-specific">
          <idmef:name>DOM race condition</idmef:name>
          <idmef:url>file://attack-info/race.html
          </idmef:url>
        </idmef:Reference>
      </idmef:Classification>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

7.4. System Policy Violation

In this example, logins are restricted to daytime hours. The alert reports a violation of this policy that occurs when a user logs in a little after 10:00 pm. Note the use of <AdditionalData> to provide information about the policy being violated.

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="bc-ds-01">
      <idmef:Node category="dns">
        <idmef:name>dialserver.example.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc72e7ef.0x00000000">
      2000-03-09T22:18:07-05:00
    </idmef:CreateTime>
    <idmef:Source ident="s01">
      <idmef:Node ident="s01-1">
```

```
          <idmef:Address category="ipv4-addr">
            <idmef:address>127.0.0.1</idmef:address>
          </idmef:Address>
        </idmef:Node>
        <idmef:Service ident="s01-2">
          <idmef:port>4325</idmef:port>
        </idmef:Service>
      </idmef:Source>
      <idmef:Target ident="t01">
        <idmef:Node ident="t01-1" category="dns">
          <idmef:name>mainframe.example.com</idmef:name>
        </idmef:Node>
        <idmef:User ident="t01-2" category="os-device">
          <idmef:UserId ident="t01-3" type="current-user">
            <idmef:name>louis</idmef:name>
            <idmef:number>501</idmef:number>
          </idmef:UserId>
        </idmef:User>
        <idmef:Service ident="t01-4">
          <idmef:name>login</idmef:name>
          <idmef:port>23</idmef:port>
        </idmef:Service>
      </idmef:Target>
      <idmef:Classification text="Login policy violation">
        <idmef:Reference origin="user-specific">
          <idmef:name>out-of-hours activity</idmef:name>
          <idmef:url>http://my.company.com/policies
          </idmef:url>
        </idmef:Reference>
      </idmef:Classification>
      <idmef:AdditionalData type="date-time"
                            meaning="start-time">
        <idmef:date-time>2000-03-09T07:00:00-05:00</idmef:date-time>
      </idmef:AdditionalData>
      <idmef:AdditionalData type="date-time"
                            meaning="stop-time">
        <idmef:date-time>2000-03-09T19:30:00-05:00</idmef:date-time>
      </idmef:AdditionalData>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

7.5. Correlated Alerts

The following example shows how the port scan alert from Section 7.2.2 could be represented if it had been detected and sent from a correlation engine, instead of a single analyzer.

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="bc-corr-01">
      <idmef:Node category="dns">
        <idmef:name>correlator01.example.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc72423b.0x00000000">
      2000-03-09T15:31:07Z
    </idmef:CreateTime>
    <idmef:Source ident="a1">
      <idmef:Node ident="a1-1">
        <idmef:Address ident="a1-2" category="ipv4-addr">
          <idmef:address>192.0.2.200</idmef:address>
        </idmef:Address>
      </idmef:Node>
    </idmef:Source>
    <idmef:Target ident="a2">
      <idmef:Node ident="a2-1" category="dns">
        <idmef:name>www.example.com</idmef:name>
        <idmef:Address ident="a2-2" category="ipv4-addr">
          <idmef:address>192.0.2.50</idmef:address>
        </idmef:Address>
      </idmef:Node>
      <idmef:Service ident="a2-3">
        <idmef:portlist>5-25,37,42,43,53,69-119,123-514
        </idmef:portlist>
      </idmef:Service>
    </idmef:Target>
    <idmef:Classification text="Portscan">
      <idmef:Reference origin="vendor-specific">
        <idmef:name>portscan</idmef:name>
        <idmef:url>http://www.vendor.com/portscan</idmef:url>
      </idmef:Reference>
    </idmef:Classification>
    <idmef:CorrelationAlert>
      <idmef:name>multiple ports in short time</idmef:name>
      <idmef:alertident>123456781</idmef:alertident>
      <idmef:alertident>123456782</idmef:alertident>
```

```
        <idmef:alertident>123456783</idmef:alertident>
        <idmef:alertident>123456784</idmef:alertident>
        <idmef:alertident>123456785</idmef:alertident>
        <idmef:alertident>123456786</idmef:alertident>
        <idmef:alertident analyzerid="a1b2c3d4">987654321
        </idmef:alertident>
        <idmef:alertident analyzerid="a1b2c3d4">987654322
        </idmef:alertident>
      </idmef:CorrelationAlert>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

7.6. Analyzer Assessments

Host-based detection of a successful unauthorized acquisition of root access through the eject buffer overflow. Note the use of <Assessment> to provide information about the analyzer's evaluation of and reaction to the attack.

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                     xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert>
    <idmef:Analyzer analyzerid="bids-192.0.2.1">
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc71e980.0x00000000">
      2000-03-09T08:12:32-01:00
    </idmef:CreateTime>
    <idmef:Source spoofed="no">
      <idmef:Node>
        <idmef:location>console</idmef:location>
        <idmef:Address category="ipv4-addr">
          <idmef:address>192.0.2.1</idmef:address>
        </idmef:Address>
      </idmef:Node>
    </idmef:Source>
    <idmef:Target decoy="no">
      <idmef:Node>
        <idmef:location>local</idmef:location>
        <idmef:Address category="ipv4-addr">
          <idmef:address>192.0.2.1</idmef:address>
        </idmef:Address>
      </idmef:Node>
      <idmef:User category="os-device">
        <idmef:UserId type="original-user">
          <idmef:number>456</idmef:number>
        </idmef:UserId>
```

```
          <idmef:UserId type="current-user">
            <idmef:name>root</idmef:name>
            <idmef:number>0</idmef:number>
          </idmef:UserId>
          <idmef:UserId type="user-privs">
            <idmef:number>0</idmef:number>
          </idmef:UserId>
        </idmef:User>
        <idmef:Process>
          <idmef:name>eject</idmef:name>
          <idmef:pid>32451</idmef:pid>
          <idmef:path>/usr/bin/eject</idmef:path>
          <idmef:arg>\x90\x80\x3f\xff...\x08/bin/sh</idmef:arg>
        </idmef:Process>
      </idmef:Target>
      <idmef:Classification
          text="Unauthorized administrative access">
        <idmef:Reference origin="vendor-specific">
          <idmef:name>Unauthorized user to superuser</idmef:name>
          <idmef:url>file://attack-info/u2s.html</idmef:url>
        </idmef:Reference>
      </idmef:Classification>
      <idmef:Assessment>
        <idmef:Impact severity="high" completion="succeeded"
                type="admin"/>
        <idmef:Action category="notification-sent">
          page
          </idmef:Action>
        <idmef:Action category="block-installed">
          disabled user (fred)
        </idmef:Action>
        <idmef:Action category="taken-offline">
          logout user (fred)
        </idmef:Action>
        <idmef:Confidence rating="high"/>
      </idmef:Assessment>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

7.7. Heartbeat

This example shows a Heartbeat message that provides "I’m alive and working" information to the manager. Note the use of <AdditionalData> elements, with "meaning" attributes, to provide some additional information.

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
                xmlns:idmef="http://iana.org/idmef">
  <idmef:Heartbeat messageid="abc123456789">
    <idmef:Analyzer analyzerid="hq-dmz-analyzer01">
      <idmef:Node category="dns">
        <idmef:location>Headquarters DMZ Network</idmef:location>
        <idmef:name>analyzer01.example.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc722ebe.0x00000000">
      2000-03-09T14:07:58Z
    </idmef:CreateTime>
    <idmef:AdditionalData type="real" meaning="%memused">
      <idmef:real>62.5</idmef:real>
    </idmef:AdditionalData>
    <idmef:AdditionalData type="real" meaning="%diskused">
      <idmef:real>87.1</idmef:real>
    </idmef:AdditionalData>
  </idmef:Heartbeat>
</idmef:IDMEF-Message>
```

7.8. XML Extension

The following example shows how to extend the IDMEF DTD. In the example, the VendorCo company has decided it wants to add geographic information to the Node class. To do this, VendorCo creates a Document Type Definition or DTD that defines how their class will be formatted:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:vendorco="http://vendor.com/idmef"
            targetNamespace="http://vendor.com/idmef"
            elementFormDefault="qualified" >

  <xsd:annotation>
    <xsd:documentation>
      Intrusion Detection Message Exchange Format (IDMEF) Extension
      for geographic information
    </xsd:documentation>
  </xsd:annotation>

  <xsd:complexType name="NodeGeoType">
    <xsd:sequence>
    <xsd:element name="latitude"
                 type="xsd:string" />
    <xsd:element name="longitude"
```

```
                    type="xsd:string" />

       <xsd:element name="elevation"
                    type="xsd:string"
                    minOccurs="0"
                    maxOccurs="1" />
           </xsd:sequence>
       <xsd:attribute name="node-ident"
                      type="xsd:integer"
                      use="required"/>
     </xsd:complexType>

     <xsd:element name="NodeGeography" type="vendorco:NodeGeoType" />

   </xsd:schema>
```

The VendorCo:NodeGeography class will contain the geographic data in three aggregate classes, VendorCo:latitude, VendorCo:longitude, and VendorCo:elevation. To associate the information in this class with a particular node, the "VendorCo:node-ident" attribute is provided; it must contain the same value as the "ident" attribute on the relevant Node element.

To make use of this DTD now, VendorCo follows the rules in Section 5.2 and defines a parameter entity called "x-vendorco" within the Document Type Definition, and then references this entity. In the alert, the VendorCo elements are included under the AdditionalData element, with a "type" attribute of "xml", as shown below.

```
<?xml version="1.0" encoding="UTF-8"?>

<idmef:IDMEF-Message version="1.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:idmef="http://iana.org/idmef"
     xmlns:vendorco="http://v.com/idmef"
     xsi:schemaLocation="http://v.com/idmef http://v.com/geo.xsd">

  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="hq-dmz-analyzer01">
      <idmef:Node category="dns">
        <idmef:location>Headquarters DMZ Network</idmef:location>
        <idmef:name>analyzer01.example.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc723b45.0xef449129">
      2000-03-09T10:01:25.93464-05:00
    </idmef:CreateTime>
```

```
      <idmef:Source ident="a1b2c3d4">
        <idmef:Node ident="a1b2c3d4-001" category="dns">
          <idmef:name>badguy.example.net</idmef:name>
          <idmef:Address ident="a1b2c3d4-002" category="ipv4-net-mask">
            <idmef:address>192.0.2.50</idmef:address>
            <idmef:netmask>255.255.255.255</idmef:netmask>
          </idmef:Address>
        </idmef:Node>
      </idmef:Source>
      <idmef:Target ident="d1c2b3a4">
        <idmef:Node ident="d1c2b3a4-001" category="dns">
          <idmef:Address category="ipv4-addr-hex">
            <idmef:address>0xde796f70</idmef:address>
          </idmef:Address>
        </idmef:Node>
      </idmef:Target>
      <idmef:Classification text="Teardrop">
        <idmef:Reference origin="bugtraqid">
          <idmef:name>124</idmef:name>
          <idmef:url>http://www.securityfocus.com/bid/124</idmef:url>
        </idmef:Reference>
      </idmef:Classification>
      <idmef:AdditionalData type="xml" meaning="node geo info">
        <idmef:xml>
          <vendorco:NodeGeography
            xmlns:vendorco="http://vendor.com/idmef"
    xsi:schemaLocation="http://v.com/idmef http://v.com/geo.xsd"
            vendorco:node-ident="a1b2c3d4-001">
          <vendorco:latitude>38.89</vendorco:latitude>
          <vendorco:longitude>-77.02</vendorco:longitude>
        </vendorco:NodeGeography>
        </idmef:xml>
      </idmef:AdditionalData>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

8. The IDMEF Document Type Definition (Normative)

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ************************************************************
 ************************************************************
 *** Intrusion Detection Message Exchange Format (IDMEF) XML DTD ***
 ***                Version 1.0, 07 March 2006                  ***
 ***                                                            ***
 *** The use and extension of the IDMEF XML DTD are described in ***
 *** RFC 4765, "The Intrusion Detection Message Exchange         ***
 *** Format",  H. Debar, D. Curry, B. Feinstein.                 ***
 ************************************************************
 ************************************************************ -->

<!-- ============================================================
 ============================================================
 === SECTION 1.  Attribute list declarations.
 ============================================================
 ============================================================ -->

<!--
 | Attributes of the IDMEF element.  In general, the fixed values of
 | these attributes will change each time a new version of the DTD
 | is released.
 -->

<!ENTITY % attlist.idmef                "
    version             CDATA                   #FIXED    '1.0'
  ">

<!--
 | Attributes of all elements.  These are the "XML" attributes that
 | every element should have.  Space handling, language, and name
 | space.
 -->
<!ENTITY % attlist.global               "
    xmlns:idmef         CDATA                   #FIXED
        'http://iana.org/idmef'
    xmlns               CDATA                   #FIXED
        'http://iana.org/idmef'
    xml:space           (default | preserve)    'default'
    xml:lang            NMTOKEN                 #IMPLIED
  ">
```

```
<!-- ================================================================
 ================================================================
 === SECTION 2.  Attribute value declarations.  Enumerated values for
 ===            many of the element-specific attribute lists.
 ================================================================
 ================================================================ -->

<!--
 | Values for the Action.category attribute.
 -->
<!ENTITY % attvals.actioncat            "
    ( block-installed | notification-sent | taken-offline | other )
  ">

<!--
 | Values for the Address.category attribute.
 -->
<!ENTITY % attvals.addrcat              "
    ( unknown | atm | e-mail | lotus-notes | mac | sna | vm |
      ipv4-addr | ipv4-addr-hex | ipv4-net | ipv4-net-mask |
      ipv6-addr | ipv6-addr-hex | ipv6-net | ipv6-net-mask )
  ">

<!--
 | Values for the AdditionalData.type attribute.
 -->
<!ENTITY % attvals.adtype               "
    ( boolean | byte | character | date-time | integer | ntpstamp |
      portlist | real | string | byte-string | xmltext )
  ">

<!--
 | Values for the Impact.completion attribute.
 -->
<!ENTITY % attvals.completion           "
    ( failed | succeeded )
  ">
<!--
 | Values for the File.category attribute.
 -->
<!ENTITY % attvals.filecat              "
    ( current | original )
  ">

<!ENTITY % attvals.fileperm "( noAccess | read | write | execute |
  search | delete | executeAs | changePermissions |
  takeOwnership)" >
```

```
   <!--
    | Values for the UserId.type attribute.
    -->
   <!ENTITY % attvals.idtype               "
       ( current-user | original-user | target-user | user-privs |
         current-group | group-privs | other-privs )
     ">

   <!--
    | Values for the Impact.type attribute.
    -->
   <!ENTITY % attvals.impacttype           "
       ( admin | dos | file | recon | user | other )
     ">

   <!--
    | Values for the Linkage.category attribute.
    -->
   <!ENTITY % attvals.linkcat              "
       ( hard-link | mount-point | reparse-point | shortcut | stream |
         symbolic-link )
     ">

   <!--
    | Values for the Checksum.algorithm attribute
    -->
   <!ENTITY % attvals.checksumalgos        "
         ( MD4 | MD5 | SHA1 | SHA2-256 | SHA2-384 | SHA2-512 | CRC-32 |
           Haval | Tiger | Gost )
     ">

   <!--
    | Values for the Node.category attribute.
    -->
   <!ENTITY % attvals.nodecat              "
       ( unknown | ads | afs | coda | dfs | dns | hosts | kerberos |
         nds | nis | nisplus | nt | wfw )
     ">

   <!--
    | Values for the Reference.origin attribute.
    -->
   <!ENTITY % attvals.origin               "
       ( unknown | vendor-specific | user-specific | bugtraqid | cve |
         osvdb )
     ">

   <!--
```

```
    | Values for the Confidence.rating attribute.
    -->
   <!ENTITY % attvals.rating               "
       ( low | medium | high | numeric )
     ">

   <!--
    | Values for the Impact.severity attribute.
    -->
   <!ENTITY % attvals.severity             "
       ( info | low | medium | high )
     ">

   <!--
    | Values for the User.category attribute.
    -->
   <!ENTITY % attvals.usercat              "
       ( unknown | application | os-device )
     ">

   <!--
    | Values for yes/no attributes such as Source.spoofed and
    | Target.decoy.
    -->
   <!ENTITY % attvals.yesno                "
       ( unknown | yes | no )
     ">

   <!-- ================================================================
    ================================================================
    === SECTION 3.  Top-level element declarations.  The IDMEF-Message
    ===            element and the types of messages it can include.
    ================================================================
    ================================================================ -->

   <!ELEMENT IDMEF-Message                 (
       (Alert | Heartbeat)*
     )>
   <!ATTLIST IDMEF-Message
       %attlist.global;
       %attlist.idmef;
     >

   <!ELEMENT Alert                         (
       Analyzer, CreateTime, DetectTime?, AnalyzerTime?,
       Source*, Target*, Classification, Assessment?, (ToolAlert |
       OverflowAlert | CorrelationAlert)?, AdditionalData*
     )>
```

```
   <!ATTLIST Alert
       messageid           CDATA                   '0'
       %attlist.global;
     >

   <!ELEMENT Heartbeat                     (
       Analyzer, CreateTime, HeartbeatInterval?, AnalyzerTime?,
       AdditionalData*
     )>
   <!ATTLIST Heartbeat
       messageid           CDATA                   '0'
       %attlist.global;
     >

   <!-- ================================================================
    ================================================================
    === SECTION 4.  Subclasses of the Alert element that provide more
    ===            data for specific types of alerts.
    ================================================================
    ================================================================ -->

   <!ELEMENT CorrelationAlert              (
       name, alertident+
     )>
   <!ATTLIST CorrelationAlert
       %attlist.global;
     >

   <!ELEMENT OverflowAlert                 (
       program, size?, buffer?
     )>
   <!ATTLIST OverflowAlert
       %attlist.global;
     >

   <!ELEMENT ToolAlert                     (
       name, command?, alertident+
     )>
   <!ATTLIST ToolAlert
       %attlist.global;
     >

   <!-- ================================================================
    ================================================================
    === SECTION 5.  The AdditionalData element.  This element allows an
    ===             alert to include additional information that cannot
    ===             be encoded elsewhere in the data model.
    ================================================================
```

```
    ============================================================== -->


   <!ELEMENT AdditionalData           (
     (boolean | byte        | character | date-time |
      integer | ntpstamp    | portlist  | real      |
      string  | byte-string | xmltext   )
    )>

   <!ATTLIST AdditionalData
       type                %attvals.adtype;        'string'
       meaning             CDATA                   #IMPLIED
       %attlist.global;
     >

   <!-- ================================================================
    ================================================================
    === SECTION 6.  Elements related to identifying entities - analyzers
    ===             (the senders of these messages), sources (of
    ===             attacks), and targets (of attacks).
    ================================================================
    ============================================================== -->

   <!ELEMENT Analyzer                      (
       Node?, Process?, Analyzer?
     )>
   <!ATTLIST Analyzer
       analyzerid          CDATA                   '0'
       name                CDATA                   #IMPLIED
       manufacturer        CDATA                   #IMPLIED
       model               CDATA                   #IMPLIED
       version             CDATA                   #IMPLIED
       class               CDATA                   #IMPLIED
       ostype              CDATA                   #IMPLIED
       osversion           CDATA                   #IMPLIED
       %attlist.global;
     >

   <!ELEMENT Classification                (
       Reference*
     )>
   <!ATTLIST Classification
       ident               CDATA                   '0'
       text                CDATA                   #REQUIRED
     >

   <!ELEMENT Source                        (
       Node?, User?, Process?, Service?
```

```
    )>
  <!ATTLIST Source
      ident               CDATA                   '0'
      spoofed             %attvals.yesno;         'unknown'
      interface           CDATA                   #IMPLIED
      %attlist.global;
    >

  <!ELEMENT Target                          (
      Node?, User?, Process?, Service?, File*
    )>
  <!ATTLIST Target
      ident               CDATA                   '0'
      decoy               %attvals.yesno;         'unknown'
      interface           CDATA                   #IMPLIED
      %attlist.global;
    >

  <!ELEMENT Assessment                      (
      Impact?, Action*, Confidence?
    )>
  <!ATTLIST Assessment
      %attlist.global;
    >

  <!-- ================================================================
   ================================================================
   === SECTION 7.  Support elements used for providing detailed info
   ===             about entities - addresses, names, etc.
   ================================================================
   ================================================================ -->

  <!ELEMENT Reference                (
      name, url
    )>
  <!ATTLIST Reference
      origin              %attvals.origin;        'unknown'
      meaning             CDATA                   #IMPLIED
    >

  <!ELEMENT Node                            (
      location?, (name | Address), Address*
    )>
  <!ATTLIST Node
      ident               CDATA                   '0'
      category            %attvals.nodecat;       'unknown'
      %attlist.global;
    >
```

```
<!ELEMENT Address                       (
    address, netmask?
  )>
<!ATTLIST Address
    ident               CDATA                   '0'
    category            %attvals.addrcat;       'unknown'
    vlan-name           CDATA                   #IMPLIED
    vlan-num            CDATA                   #IMPLIED
    %attlist.global;
  >

<!ELEMENT File                          (
    name, path, create-time?, modify-time?, access-time?,
    data-size?, disk-size?, FileAccess*, Linkage*, Inode?,
    Checksum*
  )>
<!ATTLIST File
    ident               CDATA                   '0'
    category            %attvals.filecat;       #REQUIRED
    fstype              CDATA                   #IMPLIED
    file-type           CDATA                   #IMPLIED
    %attlist.global;
  >

<!ELEMENT Permission EMPTY >
<!ATTLIST Permission
    perms               %attvals.fileperm;      #REQUIRED
    %attlist.global;
>

<!ELEMENT FileAccess                    (
    UserId, Permission+
  )>
<!ATTLIST FileAccess
    %attlist.global;
  >

<!ELEMENT Inode                         (
    change-time?, (number, major-device, minor-device)?,
    (c-major-device, c-minor-device)?
  )>
<!ATTLIST Inode
    %attlist.global;
  >

<!ELEMENT Linkage                       (
    (name, path) | File
  )>
```

```
   <!ATTLIST Linkage
       category           %attvals.linkcat;       #REQUIRED
       %attlist.global;
     >

   <!ELEMENT Checksum                    (
       value, key?
     )>
   <!ATTLIST Checksum
       algorithm          %attvals.checksumalgos; #REQUIRED
       %attlist.global;
     >

   <!ELEMENT Process                     (
       name, pid?, path?, arg*, env*
     )>
   <!ATTLIST Process
       ident              CDATA                   '0'
       %attlist.global;
     >

   <!ELEMENT Service                     (
       (((name, port?) | (port, name?)) | portlist), protocol?,
       SNMPService?, WebService?
     )>
   <!ATTLIST Service
       ident               CDATA                   '0'
   ip_version           CDATA                   #IMPLIED
   iana_protocol_number CDATA                   #IMPLIED
   iana_protocol_name   CDATA                   #IMPLIED
       %attlist.global;
     >

   <!ELEMENT SNMPService                 (
       oid?, messageProcessingModel?, securityModel?, securityName?,
       securityLevel?, contextName?, contextEngineID?, command?
     )>
   <!ATTLIST SNMPService
       %attlist.global;
     >

   <!ELEMENT User                        (
       UserId+
     )>
   <!ATTLIST User
       ident              CDATA                   '0'
       category           %attvals.usercat;       'unknown'
       %attlist.global;
```

```
    >

  <!ELEMENT UserId                        (
    (name, number?) | (number, name?)
    )>
  <!ATTLIST UserId
    ident               CDATA                   '0'
    type                %attvals.idtype;        'original-user'
    tty                 CDATA                   #IMPLIED
    %attlist.global;
    >

  <!ELEMENT WebService                    (
    url, cgi?, http-method?, arg*
    )>
  <!ATTLIST WebService
    %attlist.global;
    >

  <!-- ================================================================
   ================================================================
   === SECTION 8.  Simple elements with sub-elements or attributes of a
   ===            special nature.
   ================================================================
   ================================================================ -->

  <!ELEMENT Action              (#PCDATA) >
  <!ATTLIST Action
    category            %attvals.actioncat;     'other'
    %attlist.global;
    >

  <!ELEMENT CreateTime          (#PCDATA) >
  <!ATTLIST CreateTime
    ntpstamp            CDATA                   #REQUIRED
    %attlist.global;
    >

  <!ELEMENT DetectTime          (#PCDATA) >
  <!ATTLIST DetectTime
    ntpstamp            CDATA                   #REQUIRED
    %attlist.global;

    >

  <!ELEMENT AnalyzerTime        (#PCDATA) >
  <!ATTLIST AnalyzerTime
    ntpstamp            CDATA                   #REQUIRED
```

```
      %attlist.global;
    >

  <!ELEMENT Confidence          (#PCDATA) >
  <!ATTLIST Confidence
      rating              %attvals.rating;        'numeric'
      %attlist.global;
    >

  <!ELEMENT Impact              (#PCDATA) >
  <!ATTLIST Impact
      severity            %attvals.severity;      #IMPLIED
      completion          %attvals.completion;    #IMPLIED
      type                %attvals.impacttype;    'other'
      %attlist.global;
    >

  <!ELEMENT alertident          (#PCDATA) >
  <!ATTLIST alertident
      analyzerid          CDATA                   #IMPLIED
      %attlist.global;
    >

  <!-- ================================================================
   ================================================================
   === SECTION 9.  Simple elements with no sub-elements and no special
   === attributes.
   ================================================================
   ================================================================ -->

  <!ELEMENT boolean           (#PCDATA)         >
  <!ATTLIST boolean           %attlist.global;  >

  <!ELEMENT byte              (#PCDATA)         >
  <!ATTLIST byte              %attlist.global;  >

  <!ELEMENT character         (#PCDATA)         >
  <!ATTLIST character         %attlist.global;  >

  <!ELEMENT date-time         (#PCDATA)         >
  <!ATTLIST date-time         %attlist.global;  >

  <!ELEMENT integer           (#PCDATA)         >
  <!ATTLIST integer           %attlist.global;  >

  <!ELEMENT ntpstamp          (#PCDATA)         >
  <!ATTLIST ntpstamp          %attlist.global;  >
```

```
   <!ELEMENT real              (#PCDATA)         >
   <!ATTLIST real              %attlist.global;  >

   <!ELEMENT string            (#PCDATA)         >
   <!ATTLIST string            %attlist.global;  >

   <!ELEMENT byte-string       (#PCDATA)         >
   <!ATTLIST byte-string       %attlist.global;  >

   <!ELEMENT xmltext           ANY               >
   <!ATTLIST xmltext           %attlist.global;  >

   <!ELEMENT access-time       (#PCDATA)         >
   <!ATTLIST access-time       %attlist.global;  >

   <!ELEMENT address           (#PCDATA)         >
   <!ATTLIST address           %attlist.global;  >

   <!ELEMENT arg               (#PCDATA)         >
   <!ATTLIST arg               %attlist.global;  >

   <!ELEMENT buffer            (#PCDATA)         >
   <!ATTLIST buffer            %attlist.global;  >

   <!ELEMENT c-major-device    (#PCDATA)         >
   <!ATTLIST c-major-device    %attlist.global;  >

   <!ELEMENT c-minor-device    (#PCDATA)         >
   <!ATTLIST c-minor-device    %attlist.global;  >

   <!ELEMENT cgi               (#PCDATA)         >
   <!ATTLIST cgi               %attlist.global;  >

   <!ELEMENT change-time       (#PCDATA)         >
   <!ATTLIST change-time       %attlist.global;  >

   <!ELEMENT command           (#PCDATA)         >
   <!ATTLIST command           %attlist.global;  >

   <!ELEMENT create-time       (#PCDATA)         >
   <!ATTLIST create-time       %attlist.global;  >

   <!ELEMENT data-size         (#PCDATA)         >
   <!ATTLIST data-size         %attlist.global;  >

   <!ELEMENT disk-size         (#PCDATA)         >
   <!ATTLIST disk-size         %attlist.global;  >
```

```
   <!ELEMENT env              (#PCDATA)         >
   <!ATTLIST env              %attlist.global;  >

   <!ELEMENT http-method      (#PCDATA)         >
   <!ATTLIST http-method      %attlist.global;  >

   <!ELEMENT location         (#PCDATA)         >
   <!ATTLIST location         %attlist.global;  >

   <!ELEMENT major-device     (#PCDATA)         >
   <!ATTLIST major-device     %attlist.global;  >

   <!ELEMENT minor-device     (#PCDATA)         >
   <!ATTLIST minor-device     %attlist.global;  >

   <!ELEMENT modify-time      (#PCDATA)         >
   <!ATTLIST modify-time      %attlist.global;  >

   <!ELEMENT name             (#PCDATA)         >
   <!ATTLIST name             %attlist.global;  >

   <!ELEMENT netmask          (#PCDATA)         >
   <!ATTLIST netmask          %attlist.global;  >

   <!ELEMENT number           (#PCDATA)         >
   <!ATTLIST number           %attlist.global;  >

   <!ELEMENT oid              (#PCDATA)         >
   <!ATTLIST oid              %attlist.global;  >

   <!ELEMENT path             (#PCDATA)         >
   <!ATTLIST path             %attlist.global;  >

   <!ELEMENT permission       (#PCDATA)         >
   <!ATTLIST permission       %attlist.global;  >

   <!ELEMENT pid              (#PCDATA)         >
   <!ATTLIST pid              %attlist.global;  >

   <!ELEMENT port             (#PCDATA)         >
   <!ATTLIST port             %attlist.global;  >

   <!ELEMENT portlist         (#PCDATA)         >
   <!ATTLIST portlist         %attlist.global;  >

   <!ELEMENT program          (#PCDATA)         >
   <!ATTLIST program          %attlist.global;  >
```

```
   <!ELEMENT protocol          (#PCDATA)         >
   <!ATTLIST protocol          %attlist.global;  >

   <!ELEMENT size              (#PCDATA)         >
   <!ATTLIST size              %attlist.global;  >

   <!ELEMENT url               (#PCDATA)         >
   <!ATTLIST url               %attlist.global;  >

   <!ELEMENT HeartbeatInterval (#PCDATA)         >
   <!ATTLIST HeartbeatInterval %attlist.global;  >

   <!ELEMENT messageProcessingModel (#PCDATA)    >
   <!ATTLIST messageProcessingModel %attlist.global;>

   <!ELEMENT securityModel     (#PCDATA)         >
   <!ATTLIST securityModel     %attlist.global;  >

   <!ELEMENT securityName      (#PCDATA)         >
   <!ATTLIST securityName      %attlist.global;  >

   <!ELEMENT securityLevel     (#PCDATA)         >
   <!ATTLIST securityLevel     %attlist.global;  >

   <!ELEMENT contextName       (#PCDATA)         >
   <!ATTLIST contextName       %attlist.global;  >

   <!ELEMENT contextEngineID   (#PCDATA)         >
   <!ATTLIST contextEngineID   %attlist.global;  >

   <!ELEMENT value             (#PCDATA)         >
   <!ATTLIST value             %attlist.global;  >

   <!ELEMENT key               (#PCDATA)         >
   <!ATTLIST key               %attlist.global;  >

   <!-- End of IDMEF DTD -->
```

9. Security Considerations

This document describes a data representation for exchanging security-related information between intrusion detection system implementations. Although there are no security concerns directly applicable to the format of this data, the data itself may contain security-sensitive information whose confidentiality, integrity, and/or availability may need to be protected.

This suggests that the systems used to collect, transmit, process, and store this data should be protected against unauthorized use and that the data itself should be protected against unauthorized access. The means for achieving this protection are outside the scope of this document.

Section 5 of [2] describes the required and recommended security characteristics of the transmission protocol that will be used to deliver IDMEF data from analyzers to managers. These requirements include message confidentiality, message integrity, non-repudiation, and avoidance of duplicate messages. Both standard and proposed protocols exist that provide these features.

Where a protocol that does not meet the requirements of Section 5 of [2] is used to exchange IDMEF messages, it may be desirable to use digital signatures to certify the integrity of these messages; this is discussed in Section 6.5 of this document.

10. IANA Considerations

Section 5 describes how to use the AdditionalData class to include arbitrary "atomic" data items in an IDMEF message, as well as how AdditionalData may be used to extend the DTD itself by adding new classes and attributes.

From time to time, it may be desirable to move an extension from its private or local use status (as all extensions made via the above mechanism are) to "standard" status that should be supported by all implementations.

This may be accomplished as described in this section.

10.1. Adding Values to Existing Attributes

Several of the attributes specified in this document have lists of permissible values that they may contain. To allow the addition of new values to these lists, the IANA created a repository for attribute values called "Intrusion Detection Message Exchange Format (IDMEF) Attribute Values".

Following the policies outlined in [9], this repository is "Specification Required" by RFC. Section 10.1.1 describes the initial values for this repository.

To create a new attribute, you MUST publish an RFC to document the type. In the RFC, include a copy of the registration template found in Section 10.1.2 of this document. Put the template in your IANA Considerations section, filling in the appropriate fields. You MUST describe any interoperability and security issues in your document.

When adding a new attribute value to the repository, the IANA shall assign the next rank number in numerical sequence for the value.

10.1.1. Attribute Registrations

IDMEF Class Name: Reference

IDMEF Attribute Name: origin

Registered Values:

Rank	Keyword	Description
0	unknown	Origin of the name is not known
1	vendor-specific	A vendor-specific name (and hence, URL); this can be used to provide product-specific information
2	user-specific	A user-specific name (and hence, URL); this can be used to provide installation-specific information
3	bugtraqid	The SecurityFocus ("Bugtraq") vulnerability database identifier (http://www.securityfocus.com/bid)
4	cve	The Common Vulnerabilities and Exposures (CVE) name (http://cve.mitre.org/)
5	osvdb	The Open Source Vulnerability Database (http://www.osvdb.org)

IDMEF Class Name: Source

IDMEF Attribute Name: spoofed

Registered Values:

Rank	Keyword	Description
0	unknown	Accuracy of source information unknown
1	yes	Source is believed to be a decoy
2	no	Source is believed to be "real"

IDMEF Class Name: Target

IDMEF Attribute Name: decoy

Registered Values:

Rank	Keyword	Description
0	unknown	Accuracy of target information unknown
1	yes	Target is believed to be a decoy
2	no	Target is believed to be "real"

IDMEF Class Name: AdditionalData

IDMEF Attribute Name: type

Registered Values:

Rank	Keyword	Description
0	boolean	The element contains a boolean value, i.e., the strings "true" or "false"
1	byte	The element content is a single 8-bit byte (see Section 3.2.4)
2	character	The element content is a single character (see Section 3.2.3)
3	date-time	The element content is a date-time string (see Section 3.2.6)
4	integer	The element content is an integer (see Section 3.2.1)
5	ntpstamp	The element content is an NTP timestamp (see Section 3.2.7)
6	portlist	The element content is a list of ports (see Section 3.2.8)
7	real	The element content is a real number (see Section 3.2.2)
8	string	The element content is a string (see Section 3.2.3)
9	byte-string	The element content is a byte[] (see Section 3.2.4)
10	xmltext	The element content is XML-tagged data (see Section 5.2)

IDMEF Class Name: Impact

IDMEF Attribute Name: severity

Registered Values:

Rank	Keyword	Description
0	info	Alert represents informational activity
1	low	Low severity
2	medium	Medium severity
3	high	High severity

IDMEF Class Name: Impact

IDMEF Attribute Name: completion

Registered Values:

Rank	Keyword	Description
0	failed	The attempt was not successful
1	succeeded	The attempt succeeded

IDMEF Class Name: Impact

IDMEF Attribute Name: type

Registered Values:

Rank	Keyword	Description
0	admin	Administrative privileges were attempted or obtained
1	dos	A denial of service was attempted or completed
2	file	An action on a file was attempted or completed
3	recon	A reconnaissance probe was attempted or completed
4	user	User privileges were attempted or obtained
5	other	Anything not in one of the above categories

IDMEF Class Name: Action

IDMEF Attribute Name: category

Registered Values:

Rank	Keyword	Description
0	block-installed	A block of some sort was installed to prevent an attack from reaching its destination. The block could be a port block, address block, etc., or disabling a user account.
1	notification-sent	A notification message of some sort was sent out-of-band (via pager, e-mail, etc.). Does not include the transmission of this alert.
2	taken-offline	A system, computer, or user was taken offline, as when the computer is shut down or a user is logged off.
3	other	Anything not in one of the above categories.

IDMEF Class Name: Confidence

IDMEF Attribute Name: rating

Registered Values:

Rank	Keyword	Description
0	low	The analyzer has little confidence in its validity
1	medium	The analyzer has average confidence in its validity
2	high	The analyzer has high confidence in its validity
3	numeric	The analyzer has provided a posterior probability value indicating its confidence in its validity

IDMEF Class Name: Node

IDMEF Attribute Name: category

Registered Values:

Rank	Keyword	Description
0	unknown	Domain unknown or not relevant
1	ads	Windows 2000 Advanced Directory Services
2	afs	Andrew File System (Transarc)
3	coda	Coda Distributed File System
4	dfs	Distributed File System (IBM)
5	dns	Domain Name System
6	hosts	Local hosts file
7	kerberos	Kerberos realm
8	nds	Novell Directory Services
9	nis	Network Information Services (Sun)
10	nisplus	Network Information Services Plus (Sun)
11	nt	Windows NT domain
12	wfw	Windows for Workgroups

IDMEF Class Name: Address

IDMEF Attribute Name: category

Registered Values:

Rank	Keyword	Description
0	unknown	Address type unknown
1	atm	Asynchronous Transfer Mode network address
2	e-mail	Electronic mail address (RFC 822)
3	lotus-notes	Lotus Notes e-mail address
4	mac	Media Access Control (MAC) address
5	sna	IBM Shared Network Architecture (SNA) address
6	vm	IBM VM ("PROFS") e-mail address
7	ipv4-addr	IPv4 host address in dotted-decimal notation (a.b.c.d)
8	ipv4-addr-hex	IPv4 host address in hexadecimal notation
9	ipv4-net	IPv4 network address in dotted-decimal notation, slash, significant bits (a.b.c.d/nn)
10	ipv4-net-mask	IPv4 network address in dotted-decimal notation, slash, network mask in dotted-decimal notation (a.b.c.d/w.x.y.z)
11	ipv6-addr	IPv6 host address
12	ipv6-addr-hex	IPv6 host address in hexadecimal notation
13	ipv6-net	IPv6 network address, slash, significant bits
14	ipv6-net-mask	IPv6 network address, slash, network mask

IDMEF Class Name: User

IDMEF Attribute Name: category

Registered Values:

Rank	Keyword	Description
0	unknown	User type unknown
1	application	An application user
2	os-device	An operating system or device user

IDMEF Class Name: UserId

IDMEF Attribute Name: category

Registered Values:

Rank	Keyword	Description
0	current-user	The current user id being used by the user or process. On Unix systems, this would be the "real" user id, in general.
1	original-user	The actual identity of the user or process being reported on. On those systems that (a) do some type of auditing and (b) support extracting a user id from the "audit id" token, that value should be used. On those systems that do not support this, and where the user has logged into the system, the "login id" should be used.
2	target-user	The user id the user or process is attempting to become. This would apply, on Unix systems for example, when the user attempts to use "su", "rlogin", "telnet", etc.

	3	user-privs	Another user id the user or process has the ability to use, or a user id associated with a file permission. On Unix systems, this would be the "effective" user id in a user or process context, and the owner permissions in a file context. Multiple UserId elements of this type may be used to specify a list of privileges.
	4	current-group	The current group id (if applicable) being used by the user or process. On Unix systems, this would be the "real" group id, in general.
	5	group-privs	Another group id the group or process has the ability to use, or a group id associated with a file permission. On Unix systems, this would be the "effective" group id in a group or process context, and the group permissions in a file context. On BSD-derived Unix systems, multiple UserId elements of this type would be used to include all the group ids on the "group list".
	6	other-privs	Not used in a user, group, or process context, only used in the file context. The file permissions assigned to users who do not match either the user or group permissions on the file. On Unix systems, this would be the "world" permissions.

IDMEF Class Name: File

IDMEF Attribute Name: category

Registered Values:

Rank	Keyword	Description
0	current	The file information is from after the reported change
1	original	The file information is from before the reported change

IDMEF Class Name: File

IDMEF Attribute Name: fstype

Registered Values:

Rank	Keyword	Description
0	ufs	Berkeley Unix Fast File System
1	efs	Linux "efs" file system
2	nfs	Network File System
3	afs	Andrew File System
4	ntfs	Windows NT File System
5	fat16	16-bit Windows FAT File System
6	fat32	32-bit Windows FAT File System
7	pcfs	"PC" (MS-DOS) file system on CD-ROM
8	joliet	Joliet CD-ROM file system
9	iso9660	ISO 9660 CD-ROM file system

IDMEF Class Name: FileAccess

IDMEF Attribute Name: permission

Registered Values:

Rank	Keyword	Description
0	noAccess	No access at all is allowed for this user
1	read	This user has read access to the file
2	write	This user has write access to the file
3	execute	This user has the ability to execute the file
4	search	This user has the ability to search this file (applies to "execute" permission on directories in Unix)
5	delete	This user has the ability to delete this file
6	executeAs	This user has the ability to execute this file as another user
7	changePermissions	This user has the ability to change the access permissions on this file
8	takeOwnership	This user has the ability to take ownership of this file

IDMEF Class Name: Linkage

IDMEF Attribute Name: category

Registered Values:

Rank	Keyword	Description
0	hard-link	The <name> element represents another name for this file. This information may be more easily obtainable on NTFS file systems than others.
1	mount-point	An alias for the directory specified by the parent's <name> and <path> elements.
2	reparse-point	Applies only to Windows; excludes symbolic links and mount points, which are specific types of reparse points.
3	shortcut	The file represented by a Windows "shortcut". A shortcut is distinguished from a symbolic link because of the difference in their contents, which may be of importance to the manager.
4	stream	An Alternate Data Stream (ADS) in Windows; a fork on MacOS. Separate file system entity that is considered an extension of the main <File>.
5	symbolic-link	The <name> element represents the file to which the link points.

IDMEF Class Name: Checksum

IDMEF Attribute Name: algorithm

Registered Values:

Rank	Keyword	Description
0	MD4	The MD4 algorithm.
1	MD5	The MD5 algorithm.
2	SHA1	The SHA1 algorithm.
3	SHA2-256	The SHA2 algorithm with 256 bits length.
4	SHA2-384	The SHA2 algorithm with 384 bits length.
5	SHA2-512	The SHA2 algorithm with 512 bits length.
6	CRC-32	The CRC algorithm with 32 bits length.
7	Haval	The Haval algorithm.
8	Tiger	The Tiger algorithm.
9	Gost	The Gost algorithm.

10.1.2. Registration Template

IDMEF Class Name:

<provide the name of the class that contains the attribute to which you want to add a new value, e.g., "Address">

IDMEF Attribute Name:

<provide the name of the attribute to which you want to add a new value, e.g., "category">

New Attribute Value to Be Defined:

<provide the name of the new attribute value that you want to add, e.g., "sneaker-net">

Meaning of New Attribute Value:

<describe in detail what the attribute value means -- i.e., if an analyzer sends this value, what is it telling the receiver of the information?>

Contact Person and E-Mail Address:

<your name and e-mail address>

10.2. Adding New Attributes and Classes

To the extent possible, the IDMEF classes and attributes specified in this document have been designed to accommodate all current and near-future needs. Although it is recognized that the addition of new classes, as well as the addition of new attributes to existing classes, will be necessary in the future, these actions should not be taken lightly.

Any addition of new attributes or classes should only be undertaken when the current classes and attributes simply cannot be used to represent the information in a "clean" way -- and such additions should only be made to represent generally-useful types of data. Vendor-specific information, obscure information provided by only a particular type of analyzer or used only by a particular type of manager, "pet" attributes, and the like are not good reasons to make class and attribute additions.

At the time this RFC was written, the first anticipated case for which new classes and attributes will need to be added is to handle host-based intrusion detection systems. However, such additions should not be made until some level of consensus has been reached about the set of data that will be provided by these systems.

Following the policies outlined in [9], the addition of new classes and attributes to the IDMEF requires "IETF Consensus".

To add new attributes or classes, you MUST publish an RFC to document them, and get that RFC approved by the IESG. Typically, the IESG will seek input on prospective additions from appropriate persons (e.g., a relevant working group if one exists). You MUST describe any interoperability and security issues in your document.

11. References

11.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Wood, M. and M. Erlinger, "Intrusion Detection Mesage Exchange Requirements", RFC 4766, March 2007.

[3] Sperberg-McQueen, C., Paoli, J., Maler, E., and T. Bray, "Extensible Markup Language (XML) 1.0 (Second Edition)", World Wide Web Consortium FirstEdition http://www.w3.org/TR/2000/REC-xml-20001006, October 2000.

[4] Bray, T., Hollander, D., and A. Layman, "Namespaces in XML", World Wide Web Consortium Recommendation http://www.w3.org/TR/1999/REC-xml-names-19990114, January 1999.

[5] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[6] International Organization for Standardization, "Data elements and interchange formats - Information interchange - Representation of dates and times", ISO Standard 8601, Second Edition, December 2000.

[7] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation", RFC 1305, March 1992.

[8] Mills, D., "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI", RFC 4330, January 2006.

[9] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[10] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 4646, September 2006.

11.2. Informative References

[11] St. Johns, M., "Identification Protocol", RFC 1413, February 1993.

[12] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

[13] Eastlake, D., Reagle, J., and D. Solo, "(Extensible Markup Language) XML-Signature Syntax and Processing", RFC 3275, March 2002.

[14] Rumbaugh, J., Jacobson, I., and G. Booch, "The Unified Modeling Language Reference Model", ISBN 020130998X, 1998.

[15] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.

[16] Frye, R., Levi, D., Routhier, S., and B. Wijnen, "Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework", BCP 74, RFC 3584, August 2003.

Appendix A. Acknowledgements

The following individuals contributed substantially to this document and should be recognized for their efforts. This document would not exist without their help:

Dominique Alessandri, IBM Corporation
Spencer Allain, Teknowledge Corporation
James L. Burden, California Independent Systems Operator
Marc Dacier, IBM Corporation
Oliver Dain, MIT Lincoln Laboratory
Nicolas Delon, Prelude Hybrid IDS project
David J. Donahoo, AFIWC
Michael Erlinger, Harvey Mudd College
Reinhard Handwerker, Internet Security Systems, Inc.
Ming-Yuh Huang, The Boeing Company
Glenn Mansfield, Cyber Solutions, Inc.
Joe McAlerney, Silicon Defense
Cynthia McLain, MIT Lincoln Laboratory
Paul Osterwald, Intrusion.com
Jean-Philippe Pouzol
James Riordan, IBM Corporation
Paul Sangree, Cisco Systems
Stephane Schitter, IBM Corporation
Michael J. Slifcak, Trusted Network Technologies, Inc.
Steven R. Snapp, CyberSafe Corporation
Stuart Staniford-Chen, Silicon Defense
Michael Steiner, University of Saarland
Maureen Stillman, Nokia IP Telephony
Vimal Vaidya, AXENT
Yoann Vandoorselaere, Prelude Hybrid IDS project
Andy Walther, Harvey Mudd College
Andreas Wespi, IBM Corporation
John C. C. White, MITRE
Eric D. Williams, Information Brokers, Inc.
S. Felix Wu, University of California Davis

Appendix B. The IDMEF Schema Definition (Non-normative)

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:idmef="http://iana.org/idmef"
            targetNamespace="http://iana.org/idmef"
            elementFormDefault="qualified" >

  <xsd:annotation>
    <xsd:documentation>
      Intrusion Detection Message Exchange Format (IDMEF) Version 1.0
    </xsd:documentation>
  </xsd:annotation>

  <!-- Section 1 -->
  <!-- Omitted.  This section did namespace magic and is not
       needed with XSD validation. -->

  <!-- Section 2 -->

  <!--
    Values for the Action.category attribute.
  -->
  <xsd:simpleType name="action-category">
    <xsd:restriction base="xsd:token">
      <xsd:enumeration value="block-installed"   />
      <xsd:enumeration value="notification-sent" />
      <xsd:enumeration value="taken-offline"     />
      <xsd:enumeration value="other"             />
    </xsd:restriction>
  </xsd:simpleType>

  <!--
    Values for the Address.category attribute.
   -->
  <xsd:simpleType name="address-category">
    <xsd:restriction base="xsd:token">
      <xsd:enumeration value="unknown"       />
      <xsd:enumeration value="atm"           />
      <xsd:enumeration value="e-mail"        />
      <xsd:enumeration value="lotus-notes"   />
      <xsd:enumeration value="mac"           />
      <xsd:enumeration value="sna"           />
      <xsd:enumeration value="vm"            />
      <xsd:enumeration value="ipv4-addr"     />
      <xsd:enumeration value="ipv4-addr-hex" />
      <xsd:enumeration value="ipv4-net"      />
      <xsd:enumeration value="ipv4-net-mask" />
```

```
        <xsd:enumeration value="ipv6-addr"     />
        <xsd:enumeration value="ipv6-addr-hex" />
        <xsd:enumeration value="ipv6-net"      />
        <xsd:enumeration value="ipv6-net-mask" />
      </xsd:restriction>
    </xsd:simpleType>

    <!--
     | Values for the Impact.severity attribute.
     -->
    <xsd:simpleType name="impact-severity">
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="info" />
        <xsd:enumeration value="low" />
        <xsd:enumeration value="medium" />
        <xsd:enumeration value="high" />
      </xsd:restriction>
    </xsd:simpleType>

    <!--
      Values for the Impact.completion attribute.
     -->
    <xsd:simpleType name="impact-completion">
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="failed" />
        <xsd:enumeration value="succeeded" />
      </xsd:restriction>
    </xsd:simpleType>

    <!--
     | Values for the Impact.type attribute.
     -->
    <xsd:simpleType name="impact-type">
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="admin" />
        <xsd:enumeration value="dos"   />
        <xsd:enumeration value="file"  />
        <xsd:enumeration value="recon" />
        <xsd:enumeration value="user"  />
        <xsd:enumeration value="other" />
      </xsd:restriction>
    </xsd:simpleType>

    <!--
      Values for the File.category attribute.
     -->
    <xsd:simpleType name="file-category">
      <xsd:restriction base="xsd:token">
```

```
        <xsd:enumeration value="current"  />
        <xsd:enumeration value="original" />
      </xsd:restriction>
    </xsd:simpleType>

    <!--
      Values for the FileAccess.Permissions attribute
    -->
    <xsd:simpleType name="file-permission">
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="noAccess"/>
        <xsd:enumeration value="read"/>
        <xsd:enumeration value="write"/>
        <xsd:enumeration value="execute"/>
        <xsd:enumeration value="search" />
        <xsd:enumeration value="delete" />
        <xsd:enumeration value="executeAs" />
        <xsd:enumeration value="changePermissions" />
        <xsd:enumeration value="takeOwnership" />
      </xsd:restriction>
    </xsd:simpleType>

    <!--
      Values for the Id.type attribute.
     -->
    <xsd:simpleType name="id-type">
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="current-user"  />
        <xsd:enumeration value="original-user" />
        <xsd:enumeration value="target-user"   />
        <xsd:enumeration value="user-privs"    />
        <xsd:enumeration value="current-group" />
        <xsd:enumeration value="group-privs"   />
        <xsd:enumeration value="other-privs"   />
      </xsd:restriction>
    </xsd:simpleType>

    <!--
     | Values for the Linkage.category attribute.
     -->
    <xsd:simpleType name="linkage-category">
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="hard-link"     />
        <xsd:enumeration value="mount-point"   />
        <xsd:enumeration value="reparse-point" />
        <xsd:enumeration value="shortcut"      />
        <xsd:enumeration value="stream"        />
        <xsd:enumeration value="symbolic-link" />
```

```
      </xsd:restriction>
    </xsd:simpleType>

    <!--
      | Values for the Checksum.algorithm attribute
    -->
    <xsd:simpleType name="checksum-algorithm">
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="MD4" />
        <xsd:enumeration value="MD5" />
        <xsd:enumeration value="SHA1" />
        <xsd:enumeration value="SHA2-256" />
        <xsd:enumeration value="SHA2-384" />
        <xsd:enumeration value="SHA2-512" />
        <xsd:enumeration value="CRC-32" />
        <xsd:enumeration value="Haval" />
        <xsd:enumeration value="Tiger" />
        <xsd:enumeration value="Gost" />
      </xsd:restriction>
    </xsd:simpleType>

    <!--
     | Values for the Node.category attribute.
     -->
    <xsd:simpleType name="node-category">
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="unknown"  />
        <xsd:enumeration value="ads"      />
        <xsd:enumeration value="afs"      />
        <xsd:enumeration value="coda"     />
        <xsd:enumeration value="dfs"      />
        <xsd:enumeration value="dns"      />
        <xsd:enumeration value="hosts"    />
        <xsd:enumeration value="kerberos" />
        <xsd:enumeration value="nds"      />
        <xsd:enumeration value="nis"      />
        <xsd:enumeration value="nisplus"  />
        <xsd:enumeration value="nt"       />
        <xsd:enumeration value="wfw"      />
      </xsd:restriction>
    </xsd:simpleType>

    <!--
     | Values for the reference.origin attribute.
     -->
    <xsd:simpleType name="reference-origin">
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="unknown" />
```

```
        <xsd:enumeration value="vendor-specific" />
        <xsd:enumeration value="user-specific" />
        <xsd:enumeration value="bugtraqid" />
        <xsd:enumeration value="cve" />
        <xsd:enumeration value="osvdb" />
      </xsd:restriction>
    </xsd:simpleType>

    <!--
     | Values for the Confidence.rating attribute.
     -->
    <xsd:simpleType name="confidence-rating">
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="low"    />
        <xsd:enumeration value="medium" />
        <xsd:enumeration value="high"   />
        <xsd:enumeration value="numeric" />
      </xsd:restriction>
    </xsd:simpleType>

    <!--
     | Values for the User.category attribute.
     -->
    <xsd:simpleType name="user-category">
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="unknown"     />
        <xsd:enumeration value="application" />
        <xsd:enumeration value="os-device"   />
      </xsd:restriction>
    </xsd:simpleType>

    <!--
    / Values for the additionaldata.type attribute.
    -->
    <xsd:simpleType name="additionaldata-type">
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="boolean"     />
        <xsd:enumeration value="byte"        />
        <xsd:enumeration value="character"   />
        <xsd:enumeration value="date-time"   />
        <xsd:enumeration value="integer"     />
        <xsd:enumeration value="ntpstamp"    />
        <xsd:enumeration value="portlist"    />
        <xsd:enumeration value="real"        />
        <xsd:enumeration value="string"      />
        <xsd:enumeration value="byte-string" />
        <xsd:enumeration value="xml"         />
      </xsd:restriction>
```

```
    </xsd:simpleType>


    <!--
     | Values for yes/no attributes such as Source.spoofed and
     | Target.decoy.
     -->
    <xsd:simpleType name="yes-no-type">
      <xsd:restriction base="xsd:token">
        <xsd:enumeration value="unknown" />
        <xsd:enumeration value="yes"     />
        <xsd:enumeration value="no"      />
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="port-range">
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="[0-9]{1,5}(\-[0-9]{1,5})?"/>
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="port-list">
      <xsd:list itemType="idmef:port-range" />
    </xsd:simpleType>

    <xsd:simpleType name="ntpstamp">
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="0x[A-Fa-f0-9]{8}.0x[A-Fa-f0-9]{8}"/>
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="mime-type">
      <xsd:restriction base="xsd:string">
      </xsd:restriction>
    </xsd:simpleType>


    <!-- Section 3: Top-level element declarations.  The IDMEF-Message
         element and the types of messages it can include. -->

    <xsd:complexType name="IDMEF-Message" >
      <xsd:choice minOccurs="1" maxOccurs="unbounded">
        <xsd:element ref="idmef:Alert"     />
        <xsd:element ref="idmef:Heartbeat" />
      </xsd:choice>
      <xsd:attribute name="version" type="xsd:decimal"
                     fixed="1.0" />
    </xsd:complexType>
```

```
<xsd:element name="IDMEF-Message" type="idmef:IDMEF-Message" />

<xsd:complexType name="Alert">
  <xsd:sequence>
    <xsd:element name="Analyzer"
                 type="idmef:Analyzer" />
    <xsd:element name="CreateTime"
                 type="idmef:TimeWithNtpstamp" />
    <xsd:element name="DetectTime"
                 type="idmef:TimeWithNtpstamp"
                 minOccurs="0"
                 maxOccurs="1" />
    <xsd:element name="AnalyzerTime"
                 type="idmef:TimeWithNtpstamp"
                 minOccurs="0"
                 maxOccurs="1" />
    <xsd:element name="Source"
                 type="idmef:Source"
                 minOccurs="0"
                 maxOccurs="unbounded" />
    <xsd:element name="Target"
                 type="idmef:Target"
                 minOccurs="0"
                 maxOccurs="unbounded" />
    <xsd:element name="Classification"
                 type="idmef:Classification" />
    <xsd:element name="Assessment"
                 type="idmef:Assessment"
                 minOccurs="0"
                 maxOccurs="1" />
    <xsd:choice minOccurs="0" maxOccurs="1">
      <xsd:element name="ToolAlert"
                   type="idmef:ToolAlert" />
      <xsd:element name="OverflowAlert"
                   type="idmef:OverflowAlert" />
      <xsd:element name="CorrelationAlert"
                   type="idmef:CorrelationAlert" />
    </xsd:choice>
    <xsd:element name="AdditionalData"
                 type="idmef:AdditionalData"
                 minOccurs="0"
                 maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="messageid"
                 type="xsd:string"
                 default="0" />
</xsd:complexType>
```

```
<xsd:element name="Alert" type="idmef:Alert" />

<xsd:complexType name="Heartbeat">
  <xsd:sequence>
    <xsd:element name="Analyzer" type="idmef:Analyzer" />
    <xsd:element name="CreateTime"
                 type="idmef:TimeWithNtpstamp" />
    <xsd:element name="HeartbeatInterval"
                 type="xsd:integer"
                 minOccurs="0"
                 maxOccurs="1" />
    <xsd:element name="AnalyzerTime"
                 type="idmef:TimeWithNtpstamp"
                 minOccurs="0"
                 maxOccurs="1" />
    <xsd:element name="AdditionalData"
                 type="idmef:AdditionalData"
                 minOccurs="0"
                 maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="messageid"
                 type="xsd:string"
                 default="0" />
</xsd:complexType>

<xsd:element name="Heartbeat"
             type="idmef:Heartbeat" />

<!-- Section 4: Subclasses of the Alert class that provide
     more data for specific types of alerts. -->

<xsd:complexType name="CorrelationAlert">
  <xsd:sequence>
    <xsd:element name="name"
                 type="xsd:string" />
    <xsd:element name="alertident"
                 type="idmef:Alertident"
                 minOccurs="1"
                 maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="OverflowAlert">
  <xsd:sequence>
    <xsd:element name="program"
                 type="xsd:string" />
    <xsd:element name="size"
                 type="xsd:string" />
```

```
        <xsd:element name="buffer"
                     type="xsd:hexBinary" />
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="ToolAlert">
      <xsd:sequence>
        <xsd:element name="name"
                     type="xsd:string" />
        <xsd:element name="command"
                     type="xsd:string"
                     minOccurs="0"
                     maxOccurs="1" />
        <xsd:element name="alertident"
                     type="idmef:Alertident"
                     minOccurs="1"
                     maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>

    <!-- Section 5: The AdditionalData element.  This element allows an
         alert to include additional information that cannot be encoded
         elsewhere in the data model. -->

    <xsd:complexType name="AdditionalData">
      <xsd:choice>
        <xsd:element name="boolean"
                     type="xsd:boolean" />
        <xsd:element name="byte"
                     type="xsd:byte" />
        <xsd:element name="character">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="1"/>
              <xsd:maxLength value="1"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="date-time"
                     type="xsd:dateTime" />
        <xsd:element name="integer"
                     type="xsd:integer" />
        <xsd:element name="ntpstamp"
                     type="idmef:ntpstamp" />
        <xsd:element name="portlist"
                     type="idmef:port-list" />
        <xsd:element name="real"
                     type="xsd:decimal" />
```

```
        <xsd:element name="string"
                     type="xsd:string" />
        <xsd:element name="byte-string"
                     type="xsd:hexBinary" />
        <xsd:element name="xml"
                     type="idmef:xmltext" />
      </xsd:choice>
      <xsd:attribute name="type"
                     type="idmef:additionaldata-type" />
      <xsd:attribute name="meaning"
                     type="xsd:string" />
    </xsd:complexType>

    <!-- Section 6: Elements related to identifying entities -
         analyzers (the senders of these messages), sources (of
         attacks), and targets (of attacks). -->

    <xsd:complexType name="Analyzer">
      <xsd:sequence>
        <xsd:element name="Node"
                     type="idmef:Node"
                     minOccurs="0"
                     maxOccurs="1" />
        <xsd:element name="Process"
                     type="idmef:Process"
                     minOccurs="0"
                     maxOccurs="1" />
        <xsd:element name="Analyzer"
                     type="idmef:Analyzer"
                     minOccurs="0"
                     maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="analyzerid"
                     type="xsd:string"
                     default="0" />
      <xsd:attribute name="name"
                     type="xsd:string" />
      <xsd:attribute name="manufacturer"
                     type="xsd:string" />
      <xsd:attribute name="model"
                     type="xsd:string" />
      <xsd:attribute name="version"
                     type="xsd:string" />
      <xsd:attribute name="class"
                     type="xsd:string" />
      <xsd:attribute name="ostype"
                     type="xsd:string" />
      <xsd:attribute name="osversion"
```

```
                    type="xsd:string" />
      </xsd:complexType>

      <xsd:complexType name="Source">
        <xsd:sequence>
          <xsd:element name="Node"
                       type="idmef:Node"
                       minOccurs="0"
                       maxOccurs="1" />
          <xsd:element name="User"
                       type="idmef:User"
                       minOccurs="0"
                       maxOccurs="1" />
          <xsd:element name="Process"
                       type="idmef:Process"
                       minOccurs="0"
                       maxOccurs="1" />
          <xsd:element name="Service"
                       type="idmef:Service"
                       minOccurs="0"
                       maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="ident"
                       type="xsd:string"
                       default="0" />
        <xsd:attribute name="spoofed"
                       type="idmef:yes-no-type"
                       default="unknown" />
        <xsd:attribute name="interface"
                       type="xsd:string" />
      </xsd:complexType>

      <xsd:complexType name="Target">
        <xsd:sequence>
          <xsd:element name="Node"
                       type="idmef:Node"
                       minOccurs="0"
                       maxOccurs="1" />
          <xsd:element name="User"
                       type="idmef:User"
                       minOccurs="0"
                       maxOccurs="1" />
          <xsd:element name="Process"
                       type="idmef:Process"
                       minOccurs="0"
                       maxOccurs="1" />
          <xsd:element name="Service"
                       type="idmef:Service"
```

```
                    minOccurs="0"
                    maxOccurs="1" />
        <xsd:element name="File"
                    type="idmef:File"
                    minOccurs="0"
                    maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="ident"
                    type="xsd:string"
                    default="0" />
      <xsd:attribute name="decoy"
                    type="idmef:yes-no-type"
                    default="unknown" />
      <xsd:attribute name="interface"
                    type="xsd:string" />
    </xsd:complexType>

    <!-- Section 7: Support elements used for providing detailed info
         about entities - addresses, names, etc. -->

    <xsd:complexType name="Address">
      <xsd:sequence>
        <xsd:element name="address"
                    type="xsd:string" />
        <xsd:element name="netmask"
                    type="xsd:string"
                    minOccurs="0"
                    maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="ident"
                    type="xsd:string"
                    default="0" />
      <xsd:attribute name="category"
                    type="idmef:address-category"
                    default="unknown" />
      <xsd:attribute name="vlan-name"
                    type="xsd:string" />
      <xsd:attribute name="vlan-num"
                    type="xsd:string" />
    </xsd:complexType>

    <xsd:complexType name="Assessment">
      <xsd:sequence>
        <xsd:element name="Impact"
                    type="idmef:Impact"
                    minOccurs="0"
                    maxOccurs="1" />
        <xsd:element name="Action"
```

```
                    type="idmef:Action"
                    minOccurs="0"
                    maxOccurs="unbounded" />
        <xsd:element name="Confidence"
                    type="idmef:Confidence"
                    minOccurs="0"
                    maxOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="Reference">
      <xsd:sequence>
        <xsd:element name="name"
                    type="xsd:string" />
        <xsd:element name="url"
                    type="xsd:string" />
      </xsd:sequence>
      <xsd:attribute name="origin"
                    type="idmef:reference-origin"
                    default="unknown" />
      <xsd:attribute name="meaning"
                    type="xsd:string" />
    </xsd:complexType>

    <xsd:complexType name="Classification">
      <xsd:sequence>
        <xsd:element name="Reference"
                    type="idmef:Reference"
                    minOccurs="0"
                    maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="ident"
                    type="xsd:string"
                    default="0" />
      <xsd:attribute name="text"
                    type="xsd:string"
                    use="required" />
    </xsd:complexType>

    <xsd:complexType name="File">
      <xsd:sequence>
        <xsd:element name="name"
                    type="xsd:string" />
        <xsd:element name="path"
                    type="xsd:string" />
        <xsd:element name="create-time"
                    type="xsd:dateTime"
                    minOccurs="0"
                    maxOccurs="1" />
```

```
        <xsd:element name="modify-time"
                     type="xsd:dateTime"
                     minOccurs="0"
                     maxOccurs="1" />
        <xsd:element name="access-time"
                     type="xsd:dateTime"
                     minOccurs="0"
                     maxOccurs="1" />
        <xsd:element name="data-size"
                     type="xsd:integer"
                     minOccurs="0"
                     maxOccurs="1" />
        <xsd:element name="disk-size"
                     type="xsd:integer"
                     minOccurs="0"
                     maxOccurs="1" />
        <xsd:element name="FileAccess"
                     type="idmef:FileAccess"
                     minOccurs="0"
                     maxOccurs="unbounded" />
        <xsd:element name="Linkage"
                     type="idmef:Linkage"
                     minOccurs="0"
                     maxOccurs="unbounded" />
        <xsd:element name="Inode"
                     type="idmef:Inode"
                     minOccurs="0"
                     maxOccurs="1" />
        <xsd:element name="Checksum"
                     type="idmef:Checksum"
                     minOccurs="0"
                     maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="ident"
                     type="xsd:string"
                     default="0" />
      <xsd:attribute name="category"
                     type="idmef:file-category"
                     use="required" />
      <xsd:attribute name="fstype"
                     type="xsd:string"
                     use="required" />
      <xsd:attribute name="file-type"
                     type="idmef:mime-type" />
    </xsd:complexType>

    <xsd:complexType name="Permission">
      <xsd:attribute name="perms"
```

```
                    type="idmef:file-permission"
                    use="required" />
      </xsd:complexType>

      <xsd:complexType name="FileAccess">
        <xsd:sequence>
          <xsd:element name="UserId"
                       type="idmef:UserId" />
          <xsd:element name="permission"
                       type="idmef:Permission"
                       minOccurs="1"
                       maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="Inode">
        <xsd:sequence>
          <xsd:element name="change-time"
                       type="xsd:string"
                       minOccurs="0"
                       maxOccurs="1" />
          <xsd:sequence minOccurs="0" maxOccurs="1">
            <xsd:element name="number"
                         type="xsd:string" />
            <xsd:element name="major-device"
                         type="xsd:string" />
            <xsd:element name="minor-device"
                         type="xsd:string" />
          </xsd:sequence>
          <xsd:sequence minOccurs="0" maxOccurs="1">
            <xsd:element name="c-major-device"
                         type="xsd:string" />
            <xsd:element name="c-minor-device"
                         type="xsd:string" />
          </xsd:sequence>
        </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="Linkage">
        <xsd:choice>
          <xsd:sequence>
            <xsd:element name="name" type="xsd:string" />
            <xsd:element name="path" type="xsd:string" />
          </xsd:sequence>
          <xsd:element name="File" type="idmef:File" />
        </xsd:choice>
        <xsd:attribute name="category"
                       type="idmef:linkage-category"
```

```
                    use="required" />
      </xsd:complexType>

      <xsd:complexType name="Checksum">
        <xsd:sequence>
          <xsd:element name="value"
                       type="xsd:string" />
          <xsd:element name="key"
                       type="xsd:string"
                       minOccurs="0"
                       maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="algorithm"
                       type="idmef:checksum-algorithm"
                       use="required" />
      </xsd:complexType>

      <xsd:complexType name="Node">
        <xsd:sequence>
          <xsd:element name="location"
                       type="xsd:string"
                       minOccurs="0"
                       maxOccurs="1" />
          <xsd:choice>
            <xsd:element name="name"
                         type="xsd:string" />
            <xsd:element name="Address"
                         type="idmef:Address" />
          </xsd:choice>
          <xsd:element name="Address"
                       type="idmef:Address"
                       minOccurs="0"
                       maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="ident"
                       type="xsd:string"
                       default="0" />
        <xsd:attribute name="category"
                       type="idmef:node-category"
                       default="unknown" />
      </xsd:complexType>

      <xsd:complexType name="Process">
        <xsd:sequence>
          <xsd:element name="name"
                       type="xsd:string" />
          <xsd:element name="pid"
                       type="xsd:integer"
```

```
                    minOccurs="0"
                    maxOccurs="1" />
      <xsd:element name="path"
                    type="xsd:string"
                    minOccurs="0"
                    maxOccurs="1" />
      <xsd:element name="arg"
                    type="xsd:string"
                    minOccurs="0"
                    maxOccurs="unbounded" />
      <xsd:element name="env"
                    type="xsd:string"
                    minOccurs="0"
                    maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="ident"
                    type="xsd:string"
                    default="0" />
  </xsd:complexType>

  <xsd:complexType name="Service">
    <xsd:sequence>
      <xsd:choice>
        <xsd:sequence>
          <xsd:element name="name"
                       type="xsd:string" />
          <xsd:element name="port"
                       type="xsd:integer"
                       minOccurs="0"
                       maxOccurs="1" />
        </xsd:sequence>
        <xsd:sequence>
          <xsd:element name="port"
                       type="xsd:integer" />
          <xsd:element name="name"
                       type="xsd:string"
                       minOccurs="0"
                       maxOccurs="1" />
        </xsd:sequence>
        <xsd:element name="portlist"
                     type="idmef:port-list" />
      </xsd:choice>
      <xsd:element name="protocol"
                    type="xsd:string"
                    minOccurs="0"
                    maxOccurs="1" />
      <xsd:element name="SNMPService"
                    type="idmef:SNMPService"
```

```
                    minOccurs="0"
                    maxOccurs="1" />
        <xsd:element name="WebService"
                    type="idmef:WebService"
                    minOccurs="0"
                    maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="ident"
                    type="xsd:string"
                    default="0" />
      <xsd:attribute name="ip_version"
                    type="xsd:integer" />
      <xsd:attribute name="iana_protocol_number"
                    type="xsd:integer" />
      <xsd:attribute name="iana_protocol_name"
                    type="xsd:string" />
    </xsd:complexType>

    <xsd:complexType name="WebService">
      <xsd:sequence>
        <xsd:element name="url"
                    type="xsd:anyURI" />
        <xsd:element name="cgi"
                    type="xsd:string"
                    minOccurs="0"
                    maxOccurs="1" />
        <xsd:element name="http-method"
                    type="xsd:string"
                    minOccurs="0"
                    maxOccurs="1" />
        <xsd:element name="arg"
                    type="xsd:string"
                    minOccurs="0"
                    maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="SNMPService">
      <xsd:sequence>
        <xsd:element name="oid"
                    type="xsd:string"
                    minOccurs="0"
                    maxOccurs="1" />
        <xsd:element name="messageProcessingModel"
                    type="xsd:integer"
                    minOccurs="0"
                    maxOccurs="1" />
        <xsd:element name="securityModel"
```

```
                    type="xsd:integer"
                    minOccurs="0"
                    maxOccurs="1" />
        <xsd:element name="securityName"
                    type="xsd:string"
                    minOccurs="0"
                    maxOccurs="1" />
        <xsd:element name="securityLevel"
                    type="xsd:integer"
                    minOccurs="0"
                    maxOccurs="1" />
        <xsd:element name="contextName"
                    type="xsd:string"
                    minOccurs="0"
                    maxOccurs="1" />
        <xsd:element name="contextEngineID"
                    type="xsd:string"
                    minOccurs="0"
                    maxOccurs="1" />
        <xsd:element name="command"
                    type="xsd:string"
                    minOccurs="0"
                    maxOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="User">
      <xsd:sequence>
        <xsd:element name="UserId"
                    type="idmef:UserId"
                    minOccurs="1"
                    maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="ident"
                    type="xsd:string"
                    default="0" />
      <xsd:attribute name="category"
                    type="idmef:user-category"
                    default="unknown" />
    </xsd:complexType>

    <xsd:complexType name="UserId" >
      <xsd:choice>
        <xsd:sequence>
          <xsd:element name="name"
                      type="xsd:string" />
          <xsd:element name="number"
                      type="xsd:integer"
```

```
                      minOccurs="0"
                      maxOccurs="1" />
        </xsd:sequence>
        <xsd:sequence>
          <xsd:element name="number"
                      type="xsd:integer" />
          <xsd:element name="name"
                      type="xsd:string"
                      minOccurs="0"
                      maxOccurs="1" />
        </xsd:sequence>
      </xsd:choice>
      <xsd:attribute name="ident"
                     type="xsd:string"
                     default="0" />
      <xsd:attribute name="type"
                     type="idmef:id-type"
                     default="original-user" />
      <xsd:attribute name="tty"
                     type="xsd:string" />
    </xsd:complexType>

    <!-- Section 8: Simple elements with sub-elements or attributes
         of a special nature. -->

    <xsd:complexType name="Action">
      <xsd:simpleContent>
        <xsd:extension base="xsd:string" >
          <xsd:attribute name="category"
                         type="idmef:action-category"
                         default="other" />
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="Confidence">
      <xsd:simpleContent>
        <xsd:extension base="xsd:string" >
          <xsd:attribute name="rating"
                         type="idmef:confidence-rating"
                         use="required" />
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="TimeWithNtpstamp">
      <xsd:simpleContent>
        <xsd:extension base="xsd:dateTime">
```

```
          <xsd:attribute name="ntpstamp"
                         type="idmef:ntpstamp"
                         use="required"/>
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="Impact">
      <xsd:simpleContent>
        <xsd:extension base="xsd:string" >
          <xsd:attribute name="severity"
                         type="idmef:impact-severity" />
          <xsd:attribute name="completion"
                         type="idmef:impact-completion" />
          <xsd:attribute name="type" type="idmef:impact-type"
                         default="other" />
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="Alertident">
      <xsd:simpleContent>
        <xsd:extension base="xsd:string" >
          <xsd:attribute name="analyzerid"
                         type="xsd:string" />
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="xmltext">
      <xsd:complexContent mixed="true">
        <xsd:restriction base="xsd:anyType">
          <xsd:sequence>
            <xsd:any namespace="##other"
                     processContents="lax"
                     minOccurs="0"
                     maxOccurs="unbounded" />
          </xsd:sequence>
        </xsd:restriction>
      </xsd:complexContent>
    </xsd:complexType>

  </xsd:schema>
```

Authors' Addresses

Herve Debar
France Telecom R & D
42 Rue des Coutures
Caen 14000
FR

Phone: +33 2 31 75 92 61
EMail: herve.debar@orange-ftgroup.com
URI: http://www.francetelecom.fr/

David A. Curry
Guardian Life Insurance Company of America
7 Hanover Square, 24th Floor
New York, NY 10004
US

Phone: +1 212 919-3086
EMail: david_a_curry@glic.com
URI: http://www.glic.com/

Benjamin S. Feinstein
SecureWorks, Inc.
PO Box 95007
Atlanta, GA 30347
US

Phone: +1 404 327-6339
Email: bfeinstein@acm.org
URI: http://www.secureworks.com/

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.